The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 6, pp.305-310, Dec. 31, 2016. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2016.16.6.305
JIIBC 2016-6-38

XML의 개념적 모델로부터 논리적 모델로의 변환 기법

Transformation Methodology to Logical Model from Conceptual Model of XML

김영웅*

Young-Ung Kim*

요 약 오늘날 데이터를 표현하고 교환하는 표준 언어로 XML을 사용하고 있다. XML의 개념적 모델을 정의하기 위해서는 XML이 가지는 다양한 요소들을 다이어그램으로 표현하는 표현 규칙과 생성된 다이어그램을 논리적 모델인 XML로 변환하는 변환 알고리즘을 정의해야 한다. 본 논문은 XML의 개념적 모델로부터 논리적 모델을 생성하기 위한 변환 기법을 제안한다. 본 논문에서 사용하는 개념적 모델로는 CMXML을 이용하고, 논리적 모델로 XML 스키마를 생성한다. 이를 위해 다이어그램으로 표현되는 CMXML을 정의하고, 이 다이어그램으로부터 XML 스키마를 생성하기 위한 자료구조를 정의하고 생성규칙을 정의하고, 변환 알고리즘을 제시한다.

Abstract In these days, XML is a de facto standard language for representing and exchanging data. In order to define the conceptual model of the XML, we need to define the representation rules expressed in the diagram and propose the transformation algorithm that converts the diagram into a logical model of XML. This paper proposes a transformation methodology for generating a logical model from the conceptual model of the XML. We use CMXML as a conceptual model and generate XML schema definition as a logical model. For this, we define transformation rules and data structures for XML schema, and propose a transformation algorithm.

Key Words : XML schema, CMXML, Transformation rule, Transformation algorithm

Ⅰ. 서 론

실세계의 데이터를 설계할 때 개념적, 논리적, 물리적 설계단계를 거친다. XML[11문서를 모델링할 때 먼저 실세계의 정보를 표현하기 위해 개념적 설계를 한 후 이를 바탕으로 논리적 모델인 XML로 변환한다. XML을 개념적 단계에서 설계하기 위해서는 XML 고유의 특성인 계층적 구조(hierarchical structure)와 더불어 콘텐츠 모델(content model) 등을 표현하여야 한다[2][3].

본 논문은 XML의 개념적 모델을 논리적 모델로 변환하는 변환 기법을 제안한다. 본 논문에서는 CMXML (Conceptual Model for XML)[4]을 XML의 개념적 모델로 사용하고, 변환 알고리즘을 거쳐 생성되는 논리적 모델로는 XML 스키마 정의(XSD: XML Schema Definition)[5]를 생성한다. XML의 개념적 모델을 논리적 모델로 변환하는 절차는 다음과 같다.

① CMXML으로 개념적 설계인 다이어그램을 생성
② 변환을 위한 변환 규칙을 정의

*정회원, 한성대학교 컴퓨터공학부(교신저자)
접수일자 : 2016년 9월 22일, 수정완료 : 2016년 10월 22일
게재확정일자 : 2016년 12월 9일

Received: 22 September, 2016 / Revised: 22 October, 2016 / Accepted: 9 December, 2016
*Corresponding Author: yukim@hansung.ac.kr
Dept. of Computer Engineering, Hansung University, Korea

③ 변환 알고리즘을 정의
④ XML 스키마를 위한 자료 구조를 생성
⑤ XSD를 생성한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 변환 기법의 입력이 되는 CMXML에 대해 기술하고, 제 3장에서는 CMXML을 구성하는 각각의 구성요소들을 논리적 모델로 변환하기 위한 변환 규칙을 정의한다. 제 4장에서는 제안한 CMXML로 모델링한 개념적 모델을 논리적 모델인 XML 스키마 정의로 변환하는 변환 알고리즘과 XML 스키마 자료 구조를 기술하고. 끝으로 제 5장에서 결론 및 향후 연구과제에 대해 기술한다.

II. CMXML

CMXML은 XML을 구성하는 각 개념들을 형식적으로 정의하고, 이 개념들을 심볼을 이용하여 다이어그램으로 표현하는 XML의 개념적 설계도구이다. 그림 1은 다이어그램으로 표현한 CMXML의 개념적 설계 예를 보여준다[3].

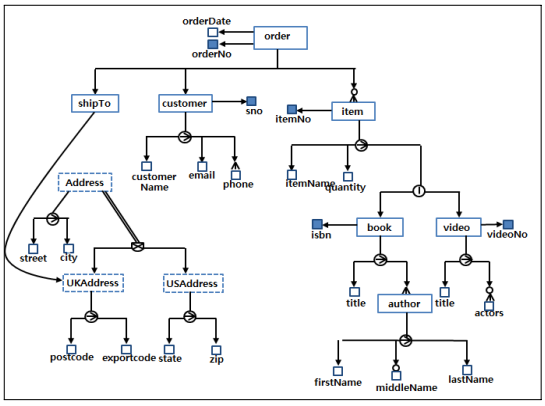


그림 1. CMXML 다이어그램
Fig. 1. CMXML Diagram

CMXML은 실세계의 객체를 표현하기 위한 노드집합, 노드들의 관계를 표현하는 방향성 에지들의 집합, 관계타입을 표현하는 라벨의 집합, 노드와 에지에 적용되는 제약조건들의 집합으로 구성된다. 노드는 비단말노드(non-terminal node)와 단말노드(terminal node)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)라 한다. 또한, 부모 노드가 없는 비단말노드를 전역 노드(global node)라 한다.

부모노드와 자식노드 사이의 에지는 generalization, aggregation, association, typeof의 종류의 타입으로 관계를 표현하며, 관계 타입이 typeof일 경우 자식노드는 추상화 노드가 되며, 자식노드가 부모노드의 타입이 된다.

노드 및 에지에서 준수해야 할 제약조건으로는 매핑 대응수(mapping cardinality), 값의 범위(최소값, 최대값), 콘텐츠 모델(sequence, all, choice 등) 등이 있다.

CMXML은 다음과 같이 심볼로 표현한다. 비단말노드는 사각형으로 표현하고, 추상화 노드의 경우 점선 사각형으로 표현한다. 단말노드는 작은 사각형으로 표현하며, 노드가 키값을 가질 경우 작은 사각형을 색을 채우고, 다중 값을 가질 경우 도메인 제약조건으로 표시하며, 리스트 값을 가질 경우 작은 사각형을 이중으로 표시하고, 내부를 채우고, 유니온 값을 가질 경우 작은 사각형 안에 세로 바로 표시한다.

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

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 분리(disjoint) 관계일 경우 반원 안에 “X”를 넣어 표시하고, 중첩(overlapping) 관계일 경우 반원 안에 작은 원을 넣어 구분한다. 관계타입이 association일 경우 두 노드사이를 점선으로 표시한다.

에지그룹에 적용되는 콘텐츠 모델은 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.

본 연구에서는 다이어그램으로 표현하기 곤란한 세부 정보들을 텍스트 형태로 표현하여 논리적 모델로의 사상을 자연스럽게 이루어지도록 하였다. 표 1은 CMXML에서 사용하는 텍스트 표현의 예를 보여준다.

표 1. CMXML 텍스트 표현
Table 1. 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

III. 변환 규칙

본 장에서는 CMXML을 구성하는 각각의 구성요소들을 논리적 스키마로 변환하기 위한 변환 규칙을 기술하고, 이 규칙에 의해 생성되는 XSD를 기술한다. 본 논문에서는 두 모델 사이의 변환 구조에 초점을 두고 있기 때문에 데이터 타입이나 도메인 무결성 제약조건 등 구조에 영향을 주지 않는 요소들에 대해서는 기술을 생략한다.

[규칙 1: 비단말노드] 비단말노드는 복합요소(complex element)로 선언한다. 복합요소는 별도의 복합타입(complex type)으로 선언하며, 타입명은 복합요소명 뒤에 *Type*을 붙여 선언한다.

CMXML	XSD
E1	<xs:element name= "E1" > <xs:complexType> ------ </xs:complexType> </xs:element>

[규칙 2: 추상화노드] 비단말노드가 추상화 노드와 association 관계타입으로 연결되어 있을 경우 비단말노드의 타입명은 추상화 노드명이 되고, 추상화 노드는 복합타입(complex type)으로 선언하며, 타입명은 추상화 노드명으로 선언한다.

CMXML	XSD
E1 T1	<xs:element name= "E1" type= "T1" /> <xs:complexType name= "T1" > ------ </xs:complexType>

[규칙 3: 단말노드] 단말노드는 부모노드에 내포하여 선언한다. 단말노드가 부모노드와 단순에지로 연결되고 노드의 타입이 ID를 가지는 경우에만 속성(attribute)으로 선언하고, 그 이외에는 단순요소(simple element)로 선언한다.

CMXML	XSD
E1 e2 e3	<xs:element name= "E1" <xs:complexType> <xs:element name= "e3" type= "xs:string" /> <xs:attribute name= "e2" type= "xs:string" /> </xs:complexType> </xs:element>

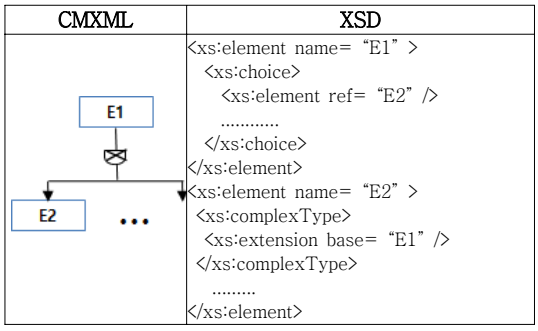
[규칙 4: 대응수 제약조건] 노드의 대응수는 최소대응수(minOccurs)와 최대대응수(maxOccurs)로 표현한다. 선언되지 않을 경우 minOccurs, maxOccurs의 값은 1이다.

CMXML	XSD
E1 E2 e3 e4	<xs:element name= "E1" > <xs:complexType> <xs:element name= "E2" minOccurs= "0" /> <xs:element name= "e3" maxOccurs= "unbounded" /> <xs:element name= "e4" minOccurs= "0" maxOccurs= "unbounded" /> </xs:complexType> </xs:element>

[규칙 5: 연관화] 관계 타입이 연관화(association)일 경우, 키 속성이 없는 노드에(두 노드가 모두 키 속성이 있다면 임의의 노드에) ref로 선언한다.

CMXML	XSD
E1 e3 E2 e4	<xs:element name= "E1" <xs:attribute name= "e3" type= "xs:ID" /> </xs:element> <xs:element name= "E2" maxOccurs= "unbounded> <xs:element ref= "E1" /> <xs:element name= "e4" type= "string" /> </xs:element>

[규칙 6: 분리 제약조건 일반화] 분리 제약조건 일반화(generalization with disjoint)는 하나의 상위 요소의 객체는 단지 하나의 하위 요소에만 속해야 하는 제약조건이다. 이 경우 일반화 관계에 참여하는 각각의 노드를 element로 선언하고, 상위 element에서 하위 element를 choice로 참조하고, 하위 element에는 상위 element를 extension으로 정의한다.



[규칙 7: 중첩 제약조건의 일반화] 중첩 제약조건의 일반화(generalization with overlapping)는 하나의 상위 요소에 속하는 개체는 다수의 하위요소의 개체에 중첩해서 속할 수 있는 제약조건이다. 중첩 제약조건의 일반화와 분리 제약조건의 일반화의 유일한 차이점은 하위 개체집합을 참조할 때 choice 선언이 없다[6][7].

IV. 변환 기법

본 장에서는 앞에서 기술한 변환 규칙을 통해 CMXML 다이어그램을 XML 스키마를 생성하는 변환 기법을 기술한다. 변환 과정은 CMXML을 통해 다이어그램을 작성하면, 이를 입력으로 제 III장에서 기술한 변환 규칙을 적용한 변환 알고리즘을 통해 XML 스키마 정보를 추출하여 XML 스키마 자료구조로 변환하고, 이를 토대로 XSD를 생성하는 과정을 거친다. 그림 2는 CMXML 다이어그램으로부터 XSD를 생성하는 과정을 보여준다.

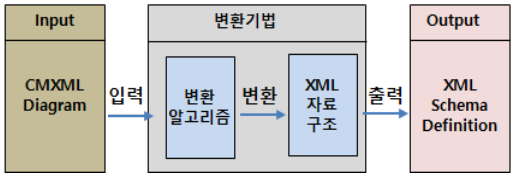


그림 2. XSD 생성 과정
Fig. 2. Process of XSD generation

1. 변환을 위한 XML 스키마 자료구조

그림 3은 변환 알고리즘을 거쳐 생성하는 XML 스키마 문서 정보를 저장하기 위한 자료구조를 보여준다. Complex Element, Simple Element는 각각 복합요소와 단순요소 정보를 저장하기 위한 자료구조이고, Attribute는 속성을 저장하기 위한 자료구조이며, Group은 요소그룹과 속성그룹을 저장하기 위한 자료구조이고, Content Model은 자식요소가 바로 콘텐츠 모델로 정의되었을 때의 정보를 저장하기 위한 자료구조이며, Extension/Restriction은 자식요소가 부모요소와 확장/축소 관계로 정의될 때의 정보를 저장하기 위한 자료구조이다.

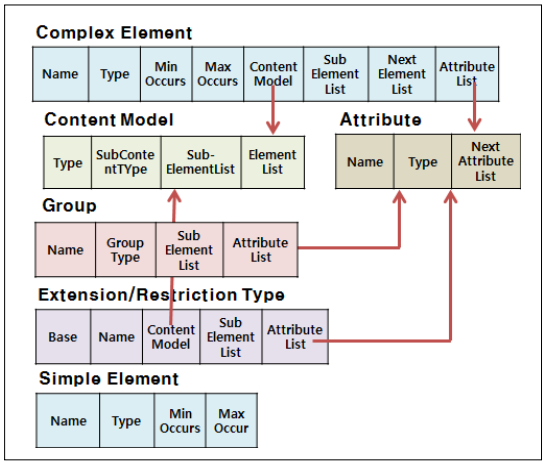


그림 3. XML 스키마 자료구조도
Fig. 3. Data structures of XML schema

2. 변환 알고리즘

본 절에서는 제 3장에서 기술한 변환 규칙을 통해 CMXML 다이어그램을 XML 스키마로 변환하는 알고리즘을 기술한다. 변환 알고리즘을 위해 표 2와 같은 기본 함수를 정의한다. 개략적인 변환 알고리즘은 표 3과 같다.

표 2. 변환 알고리즘을 위한 기본 함수
Table 2. Basic Functions for transformation algorithm

함수	기능
EnQueue(n)	노드 n을 큐에 삽입
DeQueue()	큐에서 노드를 추출, 큐가 비었으면 NULL을 반환
Parent(n)	노드 n의 부모노드를 반환, 없으면 NULL을 반환

Type(n)	노드 n의 타입을 생성, 묵시적으로는 노드명 뒤에 Type을 붙이고, n이 추상화노드와 typeof 관계(규칙 2)로 연결되어 있을 경우 추상노드명이 n의 타입명이 된다.
C(n)	노드 n의 제약조건을 생성, 제약조건은 다이어그램 또는 표 1과 같은 텍스트 표현으로부터 추출
ContentModel(n)	노드 n의 콘텐츠 모델을 생성
MakeComplexE(Parent(n), n, Type(n), C(n))	Parent(n)에 내포하여 복합요소 n을 생성
MakeSimpleE(Parent(n), n, Type(n), C(n))	Parent(n)에 내포하여 단순요소 n을 생성
MakeAtt(Parent(n), n, Type(n), C(n))	Parent(n)에 내포하여 속성 n을 생성
CreateType(n)	추상화 노드 n을 타입으로 생성

표 3. 변환 알고리즘
Table 3. Transformation algorithm

Input: CMXML Diagram

Output: XML Schema Definition

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

그림 4는 그림 1의 CMXML 다이어그램의 예를 앞에서 기술한 변환 알고리즘과 자료구조를 토대로 XML 스키마 문서정보의 자료구조를 변환한 예를 보여준다. 그림 5는 그림 4의 자료구조를 통해 생성된 최종 XSD 문서를 보여준다. 그림 5의 XSD 문서는 XML 스키마 변환 구조에 영향을 주지 않는 요소들은 생성과정에서 생략하였다.

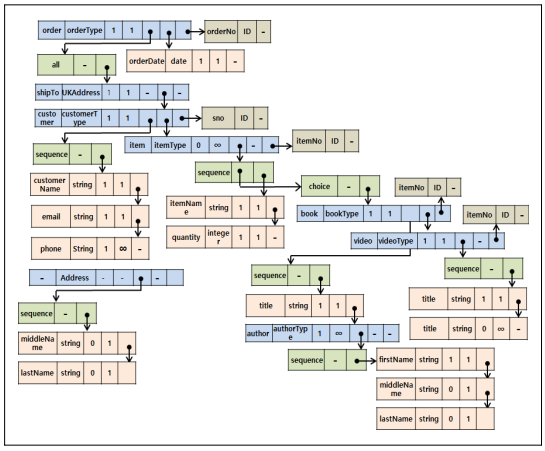


그림 4. XML 스키마 XML 스키마 자료구조
Fig. 4. Data structures of XML schema

```
<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:string/>
  </xs:sequence>
 </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:element name="order" type="orderType"/>
 <xs:sequence>
  <xs:element name="shipTo" type="UKAddress"/>
  <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:element name="orderDate" type="xs:date"/>
</xs:complexType>
</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>
<xs:complexType name="videoType">
 <xs:sequence>
 <xs:element name="title" type="xs:string"/>
 <xs:element name="actor" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="videoNo" type="xs:ID"/>
</xs:complexType>
```

그림 5. XML 스키마 문서
Fig. 5. XML schema document

V. 결 론

본 논문은 CMXML을 XML의 개념적 모델로 사용해서 논리적 모델인 XSD로 변환하는 변환 기법을 제안하였다. 변환 절차는 CMXML으로 개념적 설계인 다이어그램을 생성한다. 이를 위해 CMXML을 소개하고, 변환 규칙 및 변환 알고리즘을 정의하고, XML 스키마를 위한 자료 구조를 생성하고, 최종적으로 XSD를 생성하는 절차를 기술하였다.

본 연구의 향후 계속 진행할 연구 과제로는 본 연구의 완전성을 위해 아직 해결하지 못한 다양한 XML 개념들을 수용할 수 있는 모델로 확장하는 연구 및 논리적 모델을 개념적 모델로 역공학으로 변환하는 연구가 진행할 예정이다[8].

References

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

[2] 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.
DOI: https://doi.org/10.1145/582415.582417

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

[4] Y. Kim, “CMXML: A Conceptual Modeling Methodology for XML”, The Journal of The Institute of Webcasting, Internet and Telecommunication, VOL. 15 NO. 4, 2015.
DOI: https://doi.org/10.7236/JIIBC.2015.15.4.231

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

[6] M. Mani,“EReX: A Conceptual Model for XML”, Proceeding s of the 2nd International XML Database Symposium(XSystem 2004), 2004.
DOI: https://doi.org/10.1007/978-3-540-30081-6_10

[7] I. .H. Jung, Y. Kim, “CMXML: Extended Entity-Relationship Model for Conceptual Modeling of XML Schema”, The Journal of The Institute of Webcasting, Internet and Telecommunication, VOL. 15 NO. 1, 2015.
DOI: https://doi.org/10.7236/JIIBC.2015.15.1.157

[8] M. Necasky. “Reverse Engineering of XML Schemas to Conceptual Diagrams”, Proceeding of the 6th Asia-Pacific Conference on Conceptual Modeling, 2009.

저자 소개

김 영 웅(정회원)

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

<주관심분야 : 데이터 모델링, 소프트웨어 신뢰도, 소프트웨어 설계>