XDR 스키마를 위한 UML 클래스 다이어그램

UML Class Diagram for XDR Schema

유 문 성*
Moon-sung Yoo

요 약

XML이 웹에서 사용하는 문서와 데이터의 표준이 되고 있다. XML의 문서를 정의하는데 사용되는 것이 스키마다. 이 중 마이크로소프트사가 중심이 되어 개발하고 있는 스키마인 XDR은 스키마 중에서 실제적인 활용을 주도하고 있다. UML은 객체지향 개발 방법론에서 나온 표기법으로 시스템의 구조를 나타내는데 유용한 도구이다. 본 논문은 XDR의 스키마의 구조를 UML 클래스 다이어그램으로 변환하는 방법과 알고리즘을 제안한다. 이 변환으로 XML의 문서구조를 시각화하여 XML문서의 구조를 쉽게 파악할 수 있게 되었고 재사용성과 유연성을 높여 XML문서 개발을 효율적으로 하게 하였다.

Abstract

XML becomes the standard for exchanging documents and data on the Web. Schema is used to define XML documents. Among schema, XDR schema, developed chiefly by Microsoft, is a leading schema for practical use. UML is a notation in object-oriented software development and a useful tool to represent the structure of a system. In this paper, we study the transformation method and algorithm from XDR schema to UML diagram. By this transformation, the structure of XDR schema is represented graphically. Thus we can easily understand the structure of XDR schema and we can create XML documents effectively by enhancing reusability and flexibility.

⌐· Keyword : XML, XDR, UML, Schema

1. 서 론

최근 웹에서 사용하는 문서와 데이터의 표준을 통합하려는 시도가 XML을 중심으로 일어나 여러 분야에 적용되고 있다. XML은 기존의 HTML에서 불가능한 확장성, 구조, 검증의 특성이 있다. 즉 사용자가 원하는 대로 태그를 만들 수 있으며 태그를 구조화시킬 수 있으며 이로 만들어진 문서의 검증이 가능하다. XML은 EDI, B2B, CMS (Catalog Management System), 무선 인터넷, 홈 네트워킹, 음성인식, 국가기관의 공문서로 일부 사용되고 있으며 앞으로 이의 사용이 더욱 많은 분야로 확산될 것이다.

XML에서 태그와 문서의 구조를 사용자가 정의할 수 있는데 이에 사용되는 것이 스키마다. DTD가 기본적인 XML의 스키마로 사용되어 왔으나 XML과 다른 문법구조를 가지고 있고 표현능력이 제한되어 있는 등 여러 단점이 있어서 W3C등에서 다른 스키마를 개발하게 되었다. 이 중 마이크로소프트사가 중심이 되어 개발하고 있는 스키마인 XDR (XML-Data Reduced)은 스키마 중에서 실제적인 활용을 주도하고 있다.

XML문서의 구조를 쉽게 파악하고 체계적으로 관리하기 위해서는 XML의 문서구조를 시각화할 필요가 있다. UML은 객체지향 개발 방법론에서 나온 표기법으로 시스템의 구조를 나타내는데 유용한 도구이다.

이 논문은 XDR 스키마를 UML 클래스 다이어그램으로 변환하는 방법을 제안한다. 본 논문의 구성은 2장에서는 관련연구를 살펴보고 3장에서

* 정회원 : 상지대학 컴퓨터 정보공학부 교수
msyoo@sangji.ac.kr(제1저자)
☆ 이 논문은 2001년도 상지대학교 교내 연구비 지원에 의한 것임

XDR 스키마 문법과 UML에 대해 알아본다. 4장에서는 XDR의 UML 클래스 다이어그램으로의 변환 규칙을 제안하고 5장에서 그 규칙을 알고리즘으로 나타내었다. 6장에서 그 규칙과 알고리즘을 실례에 적용한다. 7장에서는 결론 및 향후 연구 과제를 기술하였다.

2. 관련 연구

XML문서의 구조를 여러 가지 형태로 나타내보려는 많은 시도가 있었다. 먼저 XML문서 구조를 트리형태로 나타내려는 노력으로 그 대표적인 것이 DOM(Document Object Model)[1,2]이다. 문서의 구조를 트리계층 구조로 나타냈는데 클래스의 세부적인 정보인 속성(attribute)이나 일반화 및 집단화된 관계나 제한구조를 제대로 표현하지 못했다.

다른 시도로서 XML의 여러 형식의 문서와 UML 다이어그램을 서로 변환하려는 여러 연구가 있었다. XML DTD와의 변환에 관해서는 [3]과[4], RDF 문서에 관해서는 [5], SMIL, RDF, WIDL 문서의 통합 객체 모델링에 대해서는[6], W3C 권고안에 따른 XML 스키마를 UML 클래스 다이어그램으로 변환은 [7]에서, 반대로 UML 클래스 다이어그램을 W3C 권고안에 따른 XML 스키마로 의 변환은 [8]에서 제시하였다. [9]는 SOX표준에 따라 XML 스키마를 UML 클래스 다이어그램으로 변환하였다.

이 연구 중 XML DTD와 UML의 변환[3,4]은 DTD가 EBNF 문법을 사용하여 XML문서에서 사용하는 문법과 다르며 다양한 형태의 자료형을 표현하지 못하므로 여러 가지 응용분야에 사용하기가 곤란하다. RDF, SMIL, WIDL의 변환[5,6]은 특정 응용분야 즉 메타데이터, 멀티미디어, 웹에서의 요청/응답(request/response)에 사용되는 구성요소들을 변환한 것이며 W3C 권고안에 따른 XML 스키마와의 변환[7]은 각 구성요소들을 데이터형, 요소형, 속성과 facet형으로 나누어 변환하였으나 W3C 권고안은 실제로 많이 이용되지 않는다..

본 연구는 XML Schema의 실제적인 활용을 주도하고 있는 XDR 스키마를 UML 클래스 다이어그램으로 변환하는 것이다. XDR 스키마는 자체의 고유한 구성요소인 ElementType과 AttributeType등이 있고 element와 attribute도 다른 스키마와 사용법이 다를 뿐 아니라 order 속성도 sequence, one all, many의 4가지로 W3C 권고안과 달라 이를 UML 클래스 다이어그램으로 변환하는 방법을 제안하였다.

3. XDR 스키마와 UML

3.1 XML DTD와 XML Schema

XML에서의 스키마란 문법, 구조, 어휘, 데이터 유형 등 마크업(markup)을 사용하는 방법에 관한 기술을 의미한다[10]. XML DTD(Document Type Definition)[11]도 광의의 스키마로서 XML에서 기본적으로 제공되며 구성 요소로 HTML의 태그를 생성하는 요소(Element), 속성(Attribute)과 개체(Entity)가 있다.

그러나 XML DTD(이하 DTD)는 스키마로서 여러 가지 부족한 점을 지니고 있다. 먼저 DTD는 단순한 EBNF(Extnded Bakus-Naur Form)형태의 문법을 사용하는데 이는 XML 문서 사례에서 사용되는 문법(Instance Syntax)과 다르다. 따라서 개발자들은 2종류의 다른 파서를 사용해서 개발해야 하며, 사용자들은 2종류의 문법을 배워야 되는 것이다. DTD의 표현 능력이 제한되어 있어서, 응용에 따라서 필요한 보다 많은 정보(예를 들면 정확한 반복횟수 등)의 표현이 불가능하다. DTD는 다양한 형태의 자료형(DataTypes)을 표현하지 못한다. 입학년도를 1900에서 1999의 범위만 갖도록 하는 요소를 기존의 DTD는 표현하지 못한다. 또한 이름 공간(NameSpace)이나 기본 요소값, 데이터 유형(Type), 유형 간 상속 등에 대한 지원이 불충분하거나 되지 않으며 좋은 DTD를 작성하는 것이 쉽지 않고, 기존의 DTD를 변경하여 확장할

수 있는 능력이 부족하다.

바람직한 XML 스키마는 스키마를 XML로 기술하여 XML문서에 적용하여야 한다. 이렇게 함으로써 DTD에서 나타나는 여러 가지 문제점을 해결할 수 있다. 스키마를 관리하기 위하여 DTD 파서와 같은 특별한 도구가 필요치 않고 스키마 정보를 표현하기 위하여 새로운 문법을 배울 필요가 없어진다. 스키마를 용이하게 확장 또는 제한할 수 있으며, 이름 공간 지원으로 복수의 문서에 의한 스키마 작성이 가능하다. 다양한 데이터 유형(Type)을 지원하며 검증이 가능하고, 데이터 유형 확장 및 제한에 의해서 효과적으로 내용을 재사용 할 수 있으며 이름은 같지만 내용이 다른 다중 요소를 정의할 수 있다.

그래서 W3C가 중심이 되어 적절한 스키마를 개발하고 있으며 현재 개발되었거나 진행 중인 주요 스키마는 W3C XML Schema, XML-Data Reduced(XDR), Document Content Description (DCD), Schema for Object-oriented XML (SOX), Document Definition Markup Language (DDML), Schematron Datatypes for DTDs (DT4DTD), Document Structure Description (DSD), Regular Language Description for XML (RELAX), TREX (Tree Regular Expressions for XML), RELAX, NG, Examplotron, Hook, Document Schema Definition Language (DSDL) 등이 있다. 이 중에서 W3C의 권고안인 XML Schema가 표준화를 주도하고 있으나 마이크로소프트사에서 개발하여 IE5.0이상에서 사용할 수 있는 XDR이 실제 많이 사용되고 있다.

3.2 XDR문법

XDR[12]은 마이크로소프트사가 제안한 XML-Data[13]의 기능 중 일부를 효율성을 위해서 축소한 것이다. 마이크로소프트사가 중심이 되어 개발하였으며 IBM과 같이 개발한 DCD의 영향을 받았다. IE(인터넷 익스플로러)5.0이상의 버전에서 사용할 수 있어 XML Schema의 실제적인 활용을 주도하고 있다. XML-Data는 1998년 1월 버전이 W3C에서 발표되었으나 곧 이어 1998년 7월부터 XML-Data reduced (XDR) draft가 발표되어 기존의 XML-Data를 대치하였다. 여기서는 XDR문서[12]를 근거로 하여 주요 스키마 구성 요소의 구문 구조와 특징을 기술하였다. 여기서 사용한 구문구조의 표기법은 <요소이름 {속성=속성유형(type) : 기본값(default value)}*> Content: 하위 구성요소 </요소이름> 형식이다. 용어로서 NCName은 Namespace Constraint Name의 약칭이며 Qname은 Qualified name의 약칭이다.

3.2.1 스키마(Schema)

모든 스키마는 스키마선언으로부터 시작된다. 스키마는 요소유형(ElementType), 속성유형(AttributeType), 개체(Entity)등을 구성요소로 갖는다.

```
<Schema
  mode = (open | closed) : 'open'
  name = NCName >
  Content: ((ElementType | AttributeType | Entity |
           Notation | Description)*)
</Schema>
```

3.2.2 요소 유형(ElementType)

요소 유형은 요소에서 사용할 자료형을 만들며 구문구조는 다음과 같다.

```
<ElementType
  name = NCName
  dt:type = Qname
  content= (empty | textOnly| eltOnly| mixed) : 'mixed'
  model = (open|closed) : 'open'
  order  = (seq | one | all | many) : ('seq' | 'many')>
  Content:: (datatype?, (AttributeType | ElementType)*,
           (attribute | element | group | extends)*)
</ElementType>
```

dt:type은 content=textOnly일 경우만 정의할 수 있으며 datatype은 dt:type이 없고 content=textOnly 일 경우만 정의할 수 있다. content가 eltOnly일 때

order의 기본값은 'seq'이며 content가 mixed일 때 order의 기본값은 'many'로 지정된다. 하위 구성요소로서 AttributeType 이나 ElementType을 지정했을 때 그 Type은 지역적으로만 사용된다.

3.2.3 요소(Element)

요소(element)는 태그를 만드는 구성요소이며 구문구조는 다음과 같다.

```
<element
  type= Qname
  occurs = (nonNegativeInteger':'(nonNegativeInteger |
           unbounded)?)  : '1:1' >
</element>
```

occures에서 두 번째 숫자가 생략되었을 때는 무한으로 간주한다. 이 표현을 사용하면 횟수에서 자주 사용하는 용어가 다음과 같이 나타난다.

```
required 1:1 optional  0:1 oneOrMore 1: zeroOrMore 0:
```

3.2.4 속성 유형(AttributeType)

요소에 사용되는 속성의 자료형을 만드는 구문구조는 다음과 같다

```
<AttributeType
  default = string
  dt:type = Qname  : 'string'
  name = NCName
  required= (yes | no) : 'no' >
  Content: (datatype?)
</AttributeType>
```

Content는 오직 dt:type이 생략되었을 때 가능하다.

3.2.5 속성(Attribute)

요소에 사용되는 속성을 만드는 구문구조는 다음과 같다.

```
<attribute
  default = string
  name = NCName
  required (yes | no) : 'no'
  type = Qname >
</attribute>
```

3.2.6 그룹(Group)

group은 여러 개의 구성요소들을 한단위로 처리하기 위한 것으로 그 구문구조는 다음과 같다.

```
<group
  occurs = (nonNegativeInteger':'(nonNegativeInteger |
            unbounded)?)  : '1:1'
  order =  (seq | one | all | many)>
Content: ((group | element | extends)+)
</group>
```

주변의 content가 'mixed'이면 order의 default는 'many' 이고 'eltOnly'이면 'seq'이다.

3.2.7 파생(Derivation)

extends는 이미 만들어진 유형으로부터 상속받아 새로운 구성요소를 만들기 위한 것으로 그 구문구조는 다음과 같다.

```
<extends
  type =  QName>
</extends>
```

3.2.8 개체(Entity)

개체는 문서 내에서 참조할 수 있는 문자집합의 단위로서 그 구문구조는 다음과 같다.

```
<entity
  name      NCName
 [systemID  string
  publicID  string
  notation   string]>
  Content: ((group | element | extends)+)
</entity>
```

entity가 외부에 있으면 systemID가 있으며 이때 publicID와 notation을 명기한다.

3.3 UML

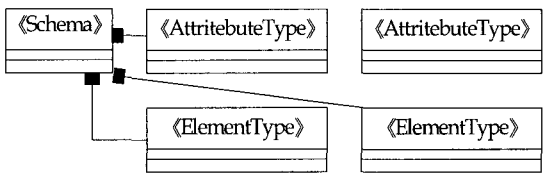
UML은 객체지향 개발 방법론에서 가장 널리 사용하고 있는 표기법으로 이 논문에서는 클래스 다이어그램을 이용한다. 클래스 다이어그램은 클래스와 클래스 사이의 관계를 나타낸 다이어그램으로 클래스는 클래스 이름, 속성과 연산들로 구성되어 있으며 관계에는 연관(association), 다중성(multiplicity), 상속(inheritance), 의존(dependency)등이 있다. 연관 중에서도 상위클래스가 하위클래스들과 부분-전체(part-whole) 관계를 가질 때 집합연관(aggregation)이라 하고 복합연관(composite)은 강한 집합연관으로서 각 컴포넌트 클래스는 오직 하나의 전체 클래스에만 속하는 관계이다[14,15].

4. XDR 스키마의 UML 클래스 다이어그램으로의 변환 규칙

XDR 스키마를 UML 클래스 다이어그램으로 변환하기 위하여 스테레오 타입과 제약을 사용하였다.

4.1 Schema

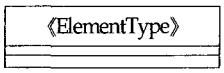
Schema는 ElementType과 AttributeType으로 구성되어 있고 이들을 UML표현으로 나타내면 다음과 같이 복합연관 관계가 된다.



(그림 1) 스키마 의 UML 다이어그램

4.2 요소 유형(ElementType)

요소 유형은 클래스가 된다. 스테레오타입 <<ElementType>>을 이용하여 표현하고 요소유형의 이름이 클래스 이름이 된다.



(그림 2) 요소유형의 UML 다이어그램

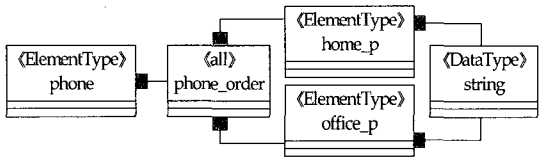
content가 textOnly일 경우 스테레오 타입이 <<DataType>>이고 클래스 이름이 dt:type인(없을 경우 string) 기본 자료형과 복합 연관 관계를 가진다. 요소유형의 order속성은 먼저 order 속성에 따른 스테레오 타입을 정의하고 각 하위 요소들과 복합 연관으로 연결한다. order속성에 따라 하위 구성요소와 다음과 같이 처리한다.

4.2.1 all

<<all>> 스테레오 타입을 사용하고 하위 요소와 단순한 복합연관이 된다.

(예)

```
<ElementType name="home_p" content="textOnly" />
<ElementType name="office_p" content="textOnly" />
<ElementType name="phone" order="all" >
  <element type="home_p"/>
  <element type="office_p"/>
</ElementType>
```



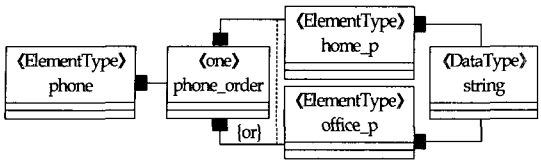
(그림 3) all의 UML 다이어그램

4.2.2 one

<<one>> 스테레오 타입을 사용하고 하위 요소들에 제약(constraint)을 주어야한다. 즉 집합연관선 사이를 모두 점선으로 연결한 후 {or} 태그를 이용하여 표기한다.

(예)

```
<ElementType name="phone" order="one" >
  <element type="home_p"/>
  <element type="office_p"/>
</ElementType>
```



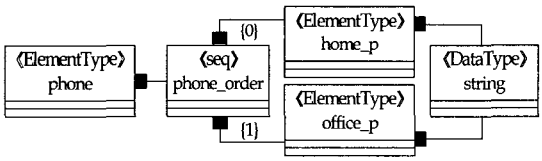
(그림 4) one의 UML 다이어그램

4.2.3 seq(sequence)

<<seq>>스테레오 타입을 사용하고 하위 요소들에 제약조건을 이용하여 하위 요소가 기술된 순서대로 숫자로 제약을 표시한다. 즉 하위 요소가 n개 있을 경우 각 요소는 기술된 순서대로 {0}부터 {n-1}까지의 값을 가지게 된다. content가 eltOnly일 때 기본값으로 지정된다.

(예)

```
<ElementType name="phone" order="seq">
  <element type="home_p"/>
  <element type="office_p"/>
</ElementType>
```



(그림 5) seq의 UML 다이어그램

4.2.4 many

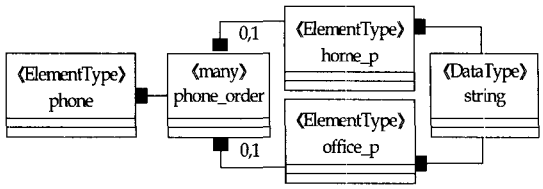
<<many>>는 하위 요소들과 다중성(multiplicity)이 0 또는 1인 관계를 갖는다. content가 mixed일 때 기본값으로 지정된다.

(예)

```
<ElementType name="phone" order="many">
  <element type="home_p"/>
  <element type="office_p"/>
</ElementType>
```

4.3 요소(Element)

요소를 하위 요소로 가진 구성요소는 요소의 유형과 복합연관 관계를 가진다. occures속성의 required(1:1)는 1로 optional(0:1)은 0,1로 oneOrMore(1:)는 1..*로 zeroOrMore(0:)는 0..*로 UML 클래스 다이어그램에서는 다중성(multiplicity)을 사용하여 나타낸다.



(그림 6) many의 UML 다이어그램

(예)

```
<ElementType name="class">
  <element type="section" occurs="oneOrMore"/>
</ElementType>
```

(그림 7) 요소의 UML 다이어그램

4.4 속성 유형(AttributeType)

속성 유형은 클래스가 된다. 스테레오타입 <<AttributeType>>을 이용하여 표현하고 요소유형의 이름이 클래스 이름이 된다. 스테레오 타입이 <<DataType>>이고 클래스 이름이 dt:type인(없을 경우 string) 기본 자료형과 복합 연관 관계를 가진다. 속성유형의 required속성이 yes,no에 따라 다중성이 1또는 0,1로 된다.

(예)

```
<AttributetType name="country" required="yes"/>
```

(그림 8) 속성유형의 UML 다이어그램

4.5 속성(Attribute)

속성을 하위 요소로 가진 구성요소는 속성의 유형과 복합연관 관계를 가진다. 속성의 required 속성이 yes, no에 따라 다중성이 1또는 0,1로 된다.

(예)

```
<ElementType name="student">
<attribute type="country" required="no"/>
</ElementType>
```

(그림 9) 속성의 UML 다이어그램

4.6 그룹(Group)

group속성은 <<group>> 스테레오 타입을 써서 나타내며 occurs는 4.2요소에 나타낸 대로 order는 4.3요소유형에 나타낸 대로 처리한다.

(예)

```
<ElementType name="a" order="all">
  <group order="many">
    <element type="b"/>
    <element type="c"/>
  </group>
</ElementType>
```

4.7 파생(Derivation)

extends속성을 하위 요소로 가진 구성요소는 extends의 type으로부터 상속을 받는다.

(예)

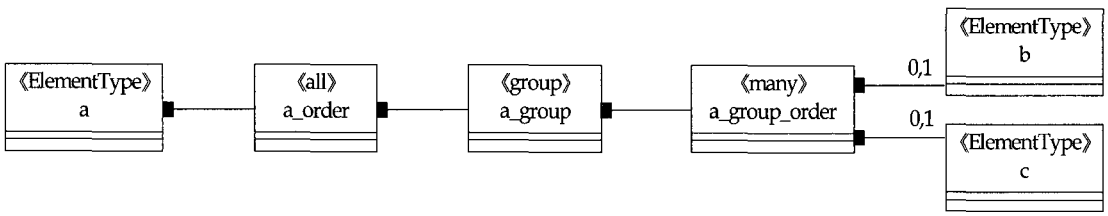
```
<ElementType name="rectangle" content="eltOnly">
  .....
  <extends type="quadrangle"/>
</ElementType>
```

(그림 11) 파생의 UML 다이어그램

5. 알고리즘

XDR 스키마를 UML 클래스 다이어그램으로 변환하기 위한 알고리즘은 다음과 같다.

```
conv(Tag tag) {
switch(tag) {
case: schema {
        create_class();
        tag=search_next_tag();
        // tag가 끝날 때까지 반복 처리
        while (!tag_empty) {
                conv(next_tag);
                tag=search_next_tag();
        }
case: elementType {
        create_class();
        // elementType의 구성요소로 element가 있을 경우
        while (!element_empty) {
          conv(element);
          // order 속성이 있을 경우 클래스끼리 order에
          // 의한 관계 연결
          if (!order_empty())
             connect_order_classes(order_type);
          // order 속성이 없을 경우 default order에 의한
          //관계 연결
             else {
```



(그림 10) group의 UML 다이어그램

```
            if (content="eltOnly")
                connect_order_classes("seq");
            else if (content="mixed")
                connect_order_classes("many");
            else;
          }
      }
      // elementType의 구성요소로 attribute가 있을 경우
      while (!attribute_empty) conv(attribute);
case: element {
      // elementType에 의해 생성된 클래스를 검색
       search_class(type);
      // occurs속성이 없으면 default가 one임
       if (occurs_empty()) occurs_type="one";
       // 클래스끼리 occurs에 의한 관계 연결
       connect_occurs_classes(occurs_type);
      }
case: attributeType {
       create_class();
       // 클래스끼리 다중성에 의한 관계 연결
       if (required) connect_occurs_classes("one");
       else  connect_occurs_classes("optional");
       }
case: attribute {
       // AttributeType에 의해 생성된 클래스를 검색
       search_class();
       if (required) connect_occur_classes("one");
       else connect_occur_classes("optional");
      }
 case: group {
      create_dummy_class();
      // group의 구성요소인 모든 element에 대하여
      while (!element_empty) {
        conv(element);
        // order 속성이 있을 경우 order에 의한 관계 연결
        if (!order_empty())
           connect_order_classes(order_type);
        // order 속성이 없을 경우 default order에 의한
        // 관계 연결
        else {
            if (content="eltOnly")
               connect_order_classes("seq");
            else if (content="mixed")
               connect_order_classes("many");
            else;
        }
      }
case: extends {
       // 확장할 elementType의 클래스 검색
       search_class(type);
       // 클래스끼리 상속성에 의한 관계 연결
       connect_extends_classes();
      }
 }
}
```

6. 변환 사례

위의 방법을 다음의 간단한 XDR Schema 문서에 적용해 보았다. 이 문서는 인사부 보고서(PernonnelReport)로서 입사일이 startDate과 endDate 사이에 있는 직원(Employee)의 이름과 주소를 가진 EmployeeReport 요소유형과 이 중 관리자(Manager)인 경우 전화번호와 email을 더 가진 ManagerReport 요소유형을 가지고 있다.

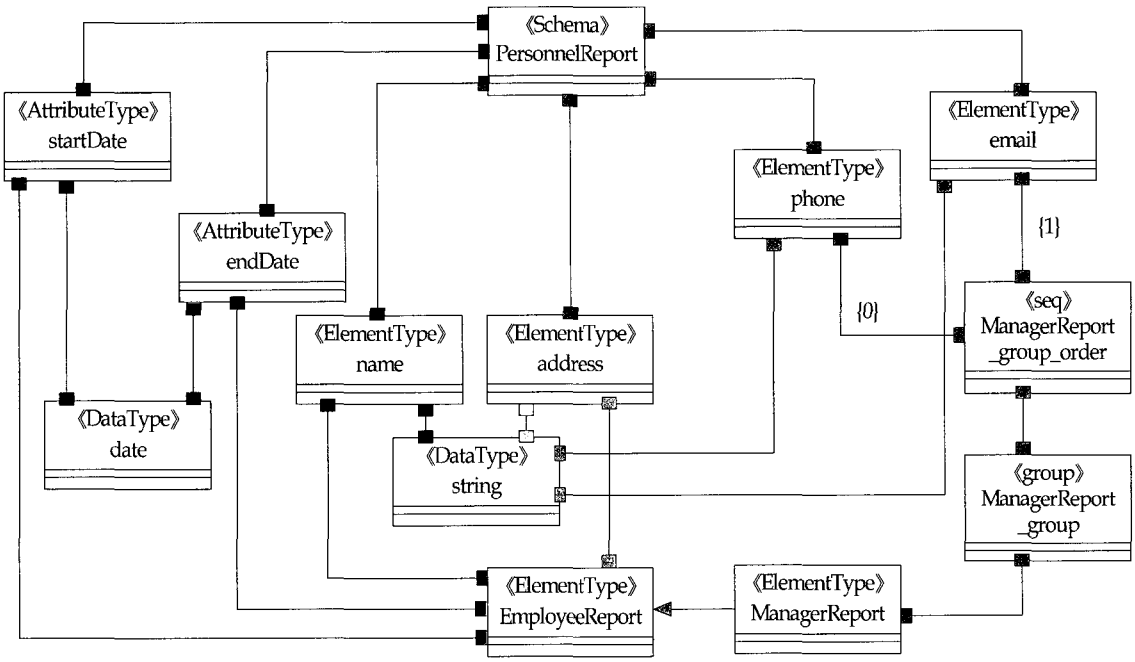
```
<Schema  name="PersonnelReport">
 <AttributeType name="startDate" dt:type="date"/>
 <AttributeType name="endDate" dt:type="date"/>
 <ElementType dt:type="string" name="name">
 </ElementType>
 <ElementType dt:type="string" name="address">
 </ElementType>
 <ElementType dt:type="string" name="phone">
 </ElementType>
 <ElementType dt:type="string" name="email">
 </ElementType>

 <ElementType name="EmployeeReport" content="eltOnly">
     <element type=name/>
     <element type=address/>
     <attribute type=startDate/>
     <attribute type=endDate/>
  </ElementType>

 <ElementType name="ManagerReport" content="eltOnly">
     <extends type="EmployeeReport"/>
     <group order="seq"/>
        <element type=phone/>
        <element type=email/>
     </group>
 </ElementType>
</Schema>
```

(그림 12) PersonnelReport의 XDR 스키마

그림 12의 스키마를 본 연구에서 제안한 알고리즘으로 UML 클래스 다이어그램으로 변환한 결



(그림 13) PersonnelReport의 UML

과가 그림 13이다.

7. 결 론

본 연구는 XDR 스키마를 UML 클래스 다이어그램으로 변환하는 방법에 대한 것이다. XDR 스키마의 문법 구성 요소인 ElementType, AttributeType, element, attribute등을 UML 클래스 다이어그램으로 변환하였으며 특히 order 속성의 all, sequence, many, one과 occurs속성의 required, optional, oneOrMore, zeroOrMore등의 변환을 중점적으로 연구하였다. 이를 알고리즘으로 기술하고 사례 연구를 통하여 그 알고리즘을 적용하였다. 이 연구를 통하여 XDR 스키마를 UML클래스 다이어그램으로 시각화함으로써 XML문서의 구조를 쉽게 파악할 수 있게 되었고 재사용성과 유연성을 높여 XML문서 개발을 효율적으로 하게 하였다. 향후 연구과제는 다른 스키마에 대하여도 UML 클래스 다이어그램으로의 변환과 이의 역변환 즉 UML 클래스 다이어그램에서 스키마로의 변환 등이 있다.

참고문헌

[1] W3C DOM WG, "Document Object Model(DOM)", http://www.w3.org/DOM/, W3C, 2002.

[2] Johnny Stenback, Andy Heninger(Editor), http://www.w3.org/TR/DOM-Level-3-Core/, W3C, 2003.

[3] 홍도석, 하얀, 김용성, "UML 클래스 다이어그램을 XML DTD로의 변환 시스템 설계 및 구현", 정보처리학회지 7권 12호, pp. 3829~3839, 2000.

[4] 채원석, 하얀, 김용성, "UML 클래스 다이어그램을 이용한 XML 문서 구조 다이어그래밍", 정보처리학회 논문지 6권 10호, pp. 2670~2679, 1999.

[5] 이미경, 하얀, 김용성, "UML 클래스 다이어그램을 이용한 RDF 스키마의 객체 모델링", 한국정보처리학회 논문지 A 7권 1호, pp. 29~40, 2000.

[6] 김상은, 하얀, 김용성, "SMIL, RDF, WIDL 문서의 통합 객체 모델링", 정보과학회 논문지 B 28권 1호, pp. 14~25, 2001.

[7] 조정길, "UML 확장 메카니즘을 이용한 XML 스키마 사상 명세", 컴퓨터산업교육기술학회 논문지 3권 2호, pp. 167~178, 2002.

[8] David Carlson, "Modeling XML Applications with UML", Addison-Wesley, 2001.

[9] Grady Booch et al., "UML for XML Schema Mapping Specification", http://www.rational.com/media/resources/media/uml_xmlschema33.pdf, 1999.

[10] Boumphrey, F. et al., "XML Applications", Wrox Press Ltd., 1998.

[11] John Cowan, "Extensible Markup Language (XML) 1.1", http://www.w3.org/TR/REC-xml11, 2002.

[12] Charles Frankston(Editor), "XML-Data Reduced", http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm, W3C, 1998.

[13] Andrew Layman(Editor), "XML-Data", http://www.w3.org/TR/1998/NOTE-XML-data/, W3C, 1998.

[14] Martin Fowler, Kendall Scott, "UML Distilled: A Brief Guide to the Standard Object Modeling Language" 2nd ed., Addison-Wesley, 1999.

[15] Grady Booch, Ivar Jacobson, James Rumbaugh, "The Unified Modeling Language User Guide", Addison-Wesley, 1998.

◐ 저 자 소 개 ◑

유 문 성

1978년 서울대학교 수학과 졸업(학사)
1991년 미국 인디아나 대학교 대학원 전산학과 졸업(석사)
1996년 미국 루이지애나대학교 대학원 전산학과 졸업(박사)
2000년~현재 : 상지대학 컴퓨터 정보공학부 교수
관심분야 : 인터넷 소프트웨어, 소프트웨어 공학, 객체지향 언어
E-mail : msyoo@sangji.ac.kr