論文
3-2-5

UML 확장 메카니즘을 이용한 XML 스키마 사상 명세 (Mapping Specification for XML Schema using UML Extension Mechanisms)

조 정 길*
(Jung-Gil Cho)

요 약

최근에 이 기종 시스템간의 구조적인 문서 교환을 위해 XML(eXtensible Markup Language)이 B2B와 각종 산업계에 급속히 확산되고 있는 시점에서 이를 모델링 하기 위한 객체 지향적인 시각화 도구가 필요하다. XML에서 현재 사용하는 문서구조 정의용 규칙인 DTD(Document Type Declaration)는 여러 산업 분야에 적용시키기가 어렵다. 이에 W3C에서 XML에 더욱 적합하고 사용자를 만족시키기에 충분한 새로운 문서 구조 정의용 규칙인 XML 스키마(Schema)의 권고안(Recommendation)을 발표하였다. 이에 XML 스키마를 객체지향 모델링 기법(UML)을 활용해 설계하면 재사용성이 높고 유연성이 좋은 문서 구조를 정의할 수가 있다. 본 논문은 XML 스키마를 UML(Unified Modeling Language)로 사상(mapping)하는 명세와 알고리즘을 제안한다.

ABSTRACT

XML documents used for exchanging structured documents between heterogeneous distributed systems are spreading among B2B and the industrial world rapidly. This condition brings an object-oriented visualization tool for modeling XML documents. It is hard to apply for DTD - the rule for declaring document type, which is employed in XML, to various industrial field. Thus, W3C announced Recommendation for XML Schema - the rule for declaring new document type which is adaptable to XML and satisfy to user. Document Type which has high reusability and flexibility can be defined owing to that XML Schema is designed by utilization of Object-Oriented-Modelling technique(UML). This document is proposed the specification and an algorithm for mapping XML Schema to UML.

1. 서론

XML은 인터넷의 중간 조직적인 전달을 위한 메타문법으로서 신속하게 표준화되어 왔다. 그래서 비지니스 분석가, 시스템 분석가, 소프트웨어 개발자들은 XML로 나타낸 정보 모델을 만들고, XML과 그것을 프로세스하는 시스템 사이의 관계를 기술하는 것이 점점 필요하게 되었다[1].

XML이 여러 산업 분야에 급속하게 확산되고 있는 시점에서 현재 사용하고 있는 문서구조 정의용 규칙인

* 정회원 : 충북대학교 컴퓨터과학과

논문접수 : 2002. 1. 22.
심사완료 : 2002. 2. 16.

DTD는 모든 분야를 만족시키기가 어렵다. 이에 W3C에서 XML에 적합하고 사용자를 만족시키기에 충분한 새로운 문서 구조 정의용 규칙인 XML 스키마의 권고안을 발표하였다[2,3,4]. XML 스키마는 문서 자체도 XML 문서이고 다양한 데이터 타입을 정의할 수 있는 장점을 가지고 있다. 이러한 XML 스키마를 이용하여 모델링 하기 위해서는 객체 지향적인 시각화 도구가 필요하다. 이에 객체 지향 모델링 기법을 활용해 XML 스키마를 설계하면 재사용성과 유연성이 좋은 문서구조를 정의할 수가 있다.

따라서 본 논문에서는 XML 스키마를 객체지향 모델링 하기 위하여 UML의 확장 매카니즘인 스테레오타입 다이어그램을 이용한 XML 스키마 다이어그램을 제안한다.

본 논문의 구성은 2장에서 관련 연구인 XML 스키마와 UML에 대하여 알아보고 3장에서는 관련연구로서 XML 문서의 모델링 방법에 대하여 알아본다. 4장에서는 UML 확장 메카니즘을 이용하여 XML 스키마 다이어그램을 생성하기 위한 사상 규칙과 알고리즘을 제안한다. 그리고 5장에서는 사상규칙에 의해 UML을 적용한 실험 및 고찰에 대하여 논의한다. 6장에서는 결론 및 향후 연구 과제에 대하여 논의한다.

2. XML 스키마 와 UML

2.1 XML 스키마

XML 스키마는 XML 문서의 내용을 제안하고 기술하는 XML 언어이다[5]. XML 스키마는 현재 W3C 권고안 단계(Recommendations phase)로 발표하였다.

XML 스키마는 DTD의 단점을 보완하기 위해 만들어 졌다. DTD는 내용 모델을 정의하는데 사용되었으며, 다음과 같이 뚜렷한 한계를 가지고 있다[5,6].

- DTD는 XML과 다른 문법을 사용한다.
- DTD는 이름공간(Namespace)을 지원하지 못한다.
- DTD는 데이터형이 제한적이다.
- DTD는 복잡하고 느슨한 확장 메카니즘을 가지고 있다.

XML 스키마는 DTD의 이런 단점을 보완하기 위하여 다음과 같은 특징을 가지고 있다.

- 45가지의 다양한 데이터형을 지원한다.
- 사용자 정의 데이터형을 선언할 수가 있다.
- 이름공간을 지원한다.
- 사용자 정의 데이터형이나 상속을 재정의 한다.
- 속성을 그룹핑한다.
- 모듈화와 재사용이라는 객체 지향적 개념을 제공한다.

2.2 XML 스키마 문법

2.2.1 엘리먼트(Element) 선언

스키마의 엘리먼트 선언은 다음의 문법형식과 같이 크게 3가지로 기술한다[7,8].

[문법형식 1]

```
<element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

[문법형식 2]

```
<element name="name" minOccurs="int" maxOccurs="int">
  <complexType>
  ...
  </complexType>
 </element>
```

[문법형식 3]

```
<element name="name" minOccurs="int" maxOccurs="int">
   <simpleType>
     <restriction base="type">
      ...
     </restriction>
     </simpleType>
   </element>
```

2.2.2 데이터형

데이터형에는 정규데이터형(Built-in datatype), simpleType, complexType이 있다. 데이터형에는 45개의형이 있으며 Facet를 이용하여 사용자 데이터형으로 만들 수가 있다.

데이터형 선언은 다음의 문법형식과 같이 기술한다.

[문법형식 1]

```
<simpleType name="name">
  <restriction base="source">
    <facet value="value"/>
     .....
  </restriction>
</simpleType>
```

[문법형식 2]

```
<complexType name="name">
  <group>
    <element name="name" type="type"/>
      .....
  </group>
</complexType>
```

2.2.3 속성(Attribute)

속성의 표현 방법은 속성이름, 데이터형, 필수/선택항목, 디폴트값 순으로 표기한다. 엘리먼트 선언 후에 속성을 선언하며, 엘리먼트의 선언과 같이 기본적인 형식과 사용자가 필요한 방식으로 재정의 할 수 있는 기능을 제공한다.

속성 선언은 다음의 문법형식과 같이 2가지로 기술한다.

[문법형식 1]

```
<attribute name="name" type="simple-type" use
="how-its-used" value="value"/>
```

[문법형식 2]

```
<attribute name="name" use="how-its-used" val
ue="value"/>
  <simpleType>
    <restriction base="simple-type">
      <facet value="value"/>
        .....
      </restriction>
    </simpleType>
</attribute>
```

2.3 UML

UML은 시스템 개발 과정에서 객체지향 시스템의 결과물을 명세화하고, 시각화하고, 문서화하기 위하여 사용되는 모델링 언어이다.

2.3.1 UML 클래스 다이어그램

클래스 다이어그램은 클래스, 인터페이스, 협력들과 그들 사이의 관계를 그림으로 표현한 것이다. XML 스키마를 객체지향 모델링하기 위해 필요한 UML 클래스 다이어그램의 구성요소들은 다음과 같다[9,10].

가. 클래스(Class)와 속성(Attribute)

클래스는 3계층의 사각형으로 그리며 이름, 속성, 오퍼레이션으로 구성된다. 속성은 클래스에 속한 특성에 이름을 붙인 것으로 이것이 가질 수 있는 값의 범위를 설정한다.

나. 관계(Relationship)

관계는 한 클래스와 다른 클래스와 연결하는 방법을 나타내며, 이에 대한 의미와 표기는 <표 1>과 같다[11].

<표 1> 관계의 의미와 표기

<Table 1> Content and Notation of Relationship

관 계	의 미	표 기
연관	클래스가 개념적으로 서로 연결	———
집합연관	전체와 부분의 관계	◇———
의존관계	한 클래스가 다른 클래스를 사용하는 관계	———>
일반화	상속의 관계(속성과 오퍼레이션)	———▶

다. 제약(Constraint)과 다중성(Multiplicity)

제약은 중괄호({ })안에 자유 형식의 텍스트가 들어있는 형태로서, 클래스가 따라야 하는 규칙을 붙여줄 때 사용한다. 두 클래스간의 연관관계가 어떠한 규칙을 따라야 하는 경우에 중괄호안의 그 제약을 연결선 부근에다가 쓰면 된다.

다중성은 연관되어 있는 두 클래스 사이에서 하나의 객체에 관련될 수 있는 다른 클래스의 객체 개수를 나타낸다. 하나의 클래스가 다른 클래스에 연관될 때 존재할 수 있는 다중성은 일대일, 일대다, 일대일 또는 그이상, 일대0 또는 1, 일대 일정범위 등이 있다. 'more'와 'many'를 표현하는 기호로서 애스터리스크(*)를 사용하며, '어느값 혹은 그이상'인 'Or'는 두개의 점(..)으로 표현한다. 표기는 '최소

발생횟수'..'최대발생횟수'로 하는데, 종류는 '0..1', '0..*', '1..*'등이 있다.

2.3.2 UML 확장 메카니즘

UML의 확장 메카니즘은 우리가 통제된 방식으로 언어를 확장할 수 있게 해준다. 여기에는 스테레오타입(Stereotype), 태그값(Tagged value), 그리고 제약 등이 있다. 태그값은 UML 구성요소의 프로퍼티를 확장시켜주는데 해당 요소의 명세 안에 새로운 정보를 만들어 낼 수가 있다. 스테레오타입은 UML의 어휘를 확장시켜 새로운 구성요소를 만들어 낼 수가 있다. 이러한 스테레오타입은 기존의 UML 요소를 기본으로 하여 다른 요소를 새로 만들 수 있게 하는 서브클래스로서 기본 클래스에 부가적인 제한 조건을 추가한 것이다. 표기는 클래스 아이콘에서 "<<"와 ">>"사이에 키워드 스트링을 쓴다.

3. 관련 연구

현재 XML 문서의 모델링에 관한 연구가 국내외적으로 활발히 연구되고 있다. SGML/XML 문서를 모델링하는 연구로는 구조도[11], XOMT 다이어그램[12], UML 클래스 다이어그램[1,13,14]을 이용하는 방법이 있다. 구조도는 트리 구조의 각 마디에 상세한 정보를 연결해 주는 트리 구조와 유사한 형태이나 엔티티나 애트리뷰트 리스트를 가지는 엘리먼트나 공유되는 부분의 복잡한 DTD를 표현하기 어렵다. XOMT 다이어그램은 객체 지향적으로 DTD를 시각화하는 모델링 방법으로 OMT를 이용하였다. 그러나 OMT 자체로 표현할 수 없는 부분이 너무 많아서 확장된 표기인 XOMT를 제안하였으나 객체지향 개념이 적절하게 사용되지 못한다. 이에 비해 UML 클래스 다이어그램은 별도의 확장된 표기법 없이도 제약조건이나 스테레오타입 등의 UML 메카니즘을 이용하여 DTD나 스키마를 시각화 할 뿐만 아니라 객체지향 개념을 사용하여 데이터베이스 스키마로의 변환을 용이하게 한다. UML 클래스 다이어그램을 이용하는 방법에서 [13]은 XML DTD를 UML 클래스 다이어그램으로 표현하였으며, [14]는 XML DTD를 UML 클래스 다이어그램을 이용하여 스키마로 변환하였으며, [1]은 SOX1.1[15] 표준에 따라 XML 스키마를 UML 클래스 다이어그램으로 표현하였다. [13][14]는 DTD를 UML로 사상하였으며 [1]은 W3C 표준 권고안을 따르지 않고 스키마를 UML로 표현하였다.

따라서 본 논문에서는 W3C 권고안 단계에 따라 XML 스키마를 UML 확장 메카니즘을 이용하여 UML로 변환하는 사상 명세를 정의하고 구현하였다. 즉, UML 확장 메카니즘을 이용하여 XML 스키마의 문법에 맞게 객체지향 개념을 사용한 클래스 다이어그램을 생성하고자 한다. 이를 위해 먼저 스키마 엘리먼트를 클래스 다이어그램 엘리먼트에 사상시키는 사상 규칙들을 정의하고, 이 규칙들에 따라 XML 스키마를 클래스 다이어그램으로 변환시키는 사상 명세를 제안한다.

4. XML 스키마의 사상 명세

XML 스키마를 UML로 사상하는데 있어서, XML 구조를 명확하게 전달하는 UML 객체의 새로운 클래스를 생성하는 데에 UML 확장 메카니즘의 스테레오타입과 제약을 사용하였다. 사상은 존재하는 UML 모델의 범위에 적용할 수가 있다. 다음과 같은 이유 때문에 UML을 확장하여 선택한다. 첫째, 확장 접근방법은 모호하지 않은 방법인 UML로서 직접적인 모델인 XML 스키마로 사용자를 고려한다. 둘째, 더 쉽게 정확한 사상을 만든다.

본 장에서는 XML 스키마로부터 UML 클래스 다이어그램으로 사상시키는 정의와 규칙을 다음과 같이 제안하고, 이를 적용시킨 결과를 제시한다. XML 스키마로부터 UML을 생성하기 위해 다음과 같은 정의와 규칙들을 제안한다.

[정의 1]

XML 문서의 엘리먼트들을 UML 스테레오 타입의 집합으로 사상한다.

4.1 데이터형의 사상 규칙

데이터형에는 정규데이터형(Built-in datatype), 정규데이터형을 기본으로 엘리먼트의 데이터형을 facet를 이용해서 생성한 simpleType, 정규데이터형과 simpleType을 확장하거나 제한하여 생성한 complexType이 있다. simpleType은 자식 엘리먼트나 속성을 가질 수 없으나, complexType은 자식 엘리먼트나 속성을 정의할 때 사용한다.

정규데이터형, simpleType, complexType의 3가지 데이터형을 적절히 사용하면 모듈화 할 수가 있는데, 이러한 모듈화는 재사용 가능하고 유연성이 좋은 스키마를 만들 수가 있다[7]. 이러한 모듈화를 하는 방법은 다음과 같다. 첫째, simpleType 타입 선언 대상을 찾는다. 둘째, complexType을 선언한다. 셋째, 루트 엘리먼트를 선언한다. 이러한 절차대로 [정의 2]에 따라 모듈화를 한다면 효율적인 스키마 제작이 가능하다.

[정의 2]

simpleType과 complexType 선언 대상 엘리먼트는 트리구조에서 하향식 방식으로 찾아내고, 스키마 작성은 최하단 엘리먼트에서 루트 엘리먼트까지 선언하는 상향식 방식으로 작성한다.

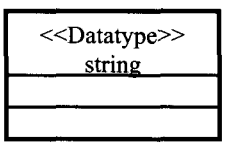
데이터형 선언에 관련된 규칙은 다음과 같다.

[규칙 1]

정규데이터형은 스테레오타입 "<<Datatype>>"라는 키워드를 쓰고 어떤 형인지를 적는다.

(스키마 소스)

```
<element name="Title" type="string"/>
```



[그림 1] [규칙 1]의 스키마 소스와 UML 표기

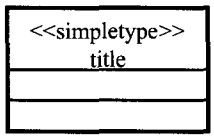
[Fig. 1] UML notation and Schema source of [Rule 1]

[규칙 2]

simpleType은 스테레오타입 "<<simpleType>>"라는 키워드를 쓰고 이름과 엘리먼트의 형을 적는다.

(스키마 소스)

```
<simpleType name="Title" >
  <restriction base="xsd:string">
   ....
  </restriction>
</simpleType>
```



[그림 2] [규칙 2]의 스키마 소스와 UML 표기

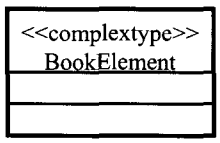
[Fig. 2] UML notation and Schema source of [Rule 2]

[규칙 3]

complexType은 스테레오타입 "<<complexType>>"라는 키워드를 쓰고 이름을 적는다.

(스키마 소스)

```
<complexType name="BookElement" >
  <sequence>
   ....
  </sequence>
</complexType>
```



[그림 3] [규칙 3]의 스키마 소스와 UML 표기

[Fig. 3] UML notation and Schema source of [Rule 3]

4.2 엘리먼트형의 사상 규칙

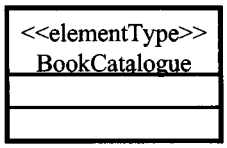
엘리먼트 선언에서 선언 부분에 오는 엘리먼트는 내용 부분에 오는 엘리먼트와 구별된다. 엘리먼트는 다른 여러 가지 데이터형을 이용하여 선언할 수 있으며, 선언에 필요한 모든 형이 정의된 후에 이루어진다. 엘리먼트 선언에 관련된 규칙은 다음과 같다.

[규칙 4]

엘리먼트 선언에서, 선언부분에 오는 엘리먼트는 클래스가 된다. 표기는 스테레오타입 "<<elementType>>"라는 키워드를 쓰고 이름을 적는다.

(스키마 소스)

```
<element name="BookCatalogue"/ >
  <complexType>
   ....
  </complexType>
</element>
```



[그림 4] [규칙 4]의 스키마 소스와 UML 표기

[Fig. 4] UML notation and Schema source of [Rule 4]

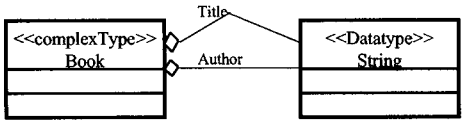
컨텐트 모델은 다른 형을 제한하거나 확장해서 생성한 complexType으로 구성한다. 컨텐트 모델은 엘리먼트 이름(name), 순서(ordinal), 반복횟수(cardinality)로 구성되어 있다. 순서는 엘리먼트가 나열된 순서를 나타낸다. 반복횟수는 minOccurs, maxOccurs의 Occurs 속성을 사용하고, 값은 0부터 n을 나타내는 unbounded까지 줄 수가 있다.

[규칙 5]

엘리먼트 내에 포함된 자식 엘리먼트형 정의는 complexType 과 데이터형 간의 집합연관 관계의 형태로 나타낸다. 엘리먼트 이름은 관계선 위에 관계 이름으로 표현한다.

(스키마 소스)

```
<complexType name="Book" >
  <sequence>
    <element name="Title" type="string"/>
    <element name="Auther" type="string"/>
     .....
  </sequence>
</complexType>
```



[그림 5] [규칙 5]의 스키마 소스와 UML 표기

[Fig. 5] UML notation and Schema source of [Rule 5]

[규칙 6]

엘리먼트 이름, 순서, 반복횟수로 구성되어 있는 컨텐트 모델에서, 순서는 제약({ })으로 표기하며 순서는 0부터 시작한다. 반복횟수는 다중성(multiplicity)으로 표현한다. 반복횟수의 표기방법은 스테레오타입에 0..1, 0..n, 1..n, 1..5등으로 한다.

그리고 컨텐트 모델은 모델 그룹을 형성한다. 이러한 모델 그룹에는 <sequence>,<choice>, <all>의 세 가지가 있다. <sequence>는 그룹의 자식 엘리먼트들이 표기한 순서대로 사용되며, <choice>는 그룹의 자식 엘리먼트들 중에서 오직 하나의 자식 엘리먼트만 사용되며, <all>은 그룹의 자식 엘리먼트들이 순서에 관계없이 사용되어진다.

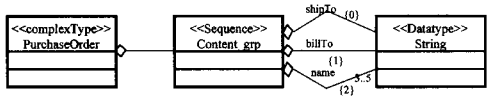
컨텐트 모델 그룹에서 <sequence>,<choice>, <all>은 순서표기를 이용한다. 이런 모델 그룹을 [규칙 7], [규칙 8], [규칙 9], [규칙 10]에서 그룹 스테레오타입으로 표기함으로서 가독성을 높였다.

[규칙 7]

<sequence>는 중괄호({ })를 이용하여 자식 엘리먼트가 기술된 순서대로 숫자 0부터 제약으로 표기하며, 그룹 스테레오타입을 정의하고 거기에서 다른 엘리먼트를 갖는다.

(스키마 소스)

```
<complexType name="PurchaseOrder" >
  <sequence>
    <element name="shipTo" type="string"/>
    <element name="billTo" type="string"/>
    <element name="name" type="string"
      minOccurs="3" maxOccurs="5" />
  </sequence>
</complexType>
```



[그림 6] [규칙 6][규칙 7]의 스키마 소스와 UML 표기

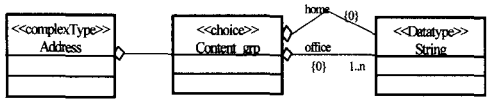
[Fig. 6] UML notation and Schema source of [Rule 6][Rule 7]

[규칙 8]

<choice>는 관계된 엘리먼트들을 모두 같은 번호로 중괄호({ })를 이용하여 표기하며, 그룹 스테레오타입을 정의하고 거기에서 다른 엘리먼트를 갖는다.

(스키마 소스)

```
<complexType name="Address" >
  <choice>
    <element name="home" type="string"/>
    <element name="office" type="string"
      maxOccurs="unbounded" />
     .....
  </choice>
</complexType>
```



[그림 7] [규칙 8]의 스키마 소스와 UML 표기

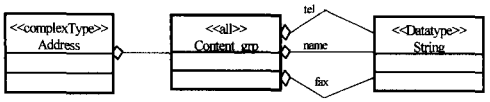
[Fig. 7] UML notation and Schema source of [Rule 8]

[규칙 9]

<all>은 중괄호({ })를 이용하여 모든 엘리먼트가 순서 제약이 없이 표기하며, 그룹 스테레오타입을 정의하고 거기에서 다른 엘리먼트를 갖는다.

(스키마 소스)

```
<complexType name="Address" >
  <all>
    <element name="name" type="string"/>
    <element name="tel" type="string" />
    <element name="fax" type="string" />
   .....
  </all>
</complexType>
```



[그림 8] [규칙 9]의 스키마 소스와 UML 표기

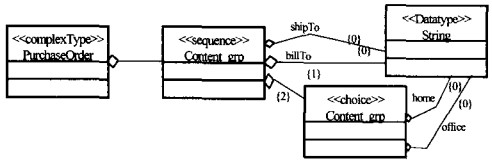
[Fig. 8] UML notation and Schema source of [Rule 9]

[규칙 10]

그룹 <sequence>, <choice>, <all>의 중복 구조 정의에서 안의 그룹 표현은 [규칙7], [규칙 8], [규칙 9]와 같은 방법으로 표기하며, 밖의 그룹 표현은 스테레오타입으로 표기한다.

(스키마 소스)

```
<complexType name="PurchaseOrder" >
  <sequence>
    <element name="shipTo" type="string"/>
    <element name="billTo" type="string" />
    <choice>
      <element name="home" type="string" />
      <element name="office" type="string" />
    </choice>
  </sequence>
</complexType>
```



[그림 9] [규칙 10]의 스키마 소스와 UML 표기

[Fig. 9] UML notation and Schema source of [Rule 10]

4.3 속성과 Facet의 사상 규칙

엘리먼트는 속성을 가질 수 있으며, 엘리먼트 선언 후에 따라온다. 또한 엘리먼트의 선언과 같이 기본적인 형식과 사용자가 필요한 방식으로 데이터형을 재정의 할 수 있는 기능을 제공한다. 속성의 표현 방법은 속성이름(name), 데이터형(type), 필수/선택항목(use), 디폴트값(value)순이다. 필수/선택항목은 local 속성 선언에만 사용하며 <implied>, <required>, <default>, <fixed>, <optional>, <prohibited>등이 있고, <default>와 <fixed>만이 디폴트값을 쓸 수가 있다.

[규칙 11]

속성의 UML 표기는 단방향 직선(Association)으로 표현하며, 이름은 관계이름으로, 필수/선택항목은 관계의 제약으로 나타낸다.

W3C의 정규 데이터형의 각각은 Facet라는 속성을 가진다. 각각의 데이터형은 각자의 Facet를 가지고 있으며, Facet를 이용해 사용자의 새로운 데이터형을 만들 수가 있다. 15개의 Facet중에서

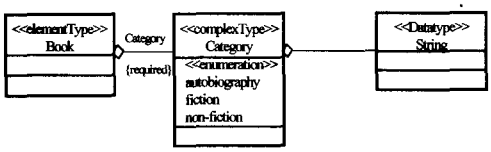
enumeration 과 pattern은 "Or"의 여러 개를 사용할 수 있는 선택 관계이고, 나머지는 한번씩만 사용할 수 있는 "And"의 관계에 있다.

[규칙 12]

Facet중에서 enumeration 과 pattern은 스테레오타입 <<enumeration>>과 <<pattern>>을 사용하여 클래스의 속성을 구분 지어 표기한다.

(스키마 소스)

```
<element name="Book" >
  <complexType>
    .....
    <attribute ref="Category" use="required"
/>
  </complexType>
</element>
<attribute nane="Category">
  <simpleType>
    <restriction base="xsd:string">
      <enumeration value="autobiography"/>
      <enumeration value="fiction"/>
      <enumeration value=non-fiction"/>
    </restriction>
  </simpleType>
</sequence>
</attribute>
```



[그림 10] [규칙 11][규칙 12]의 스키마 소스와 UML 표기

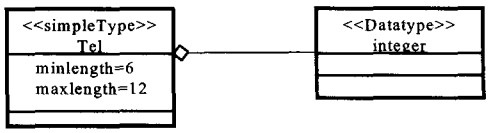
[Fig. 10] UML notation and Schema source of [Rule 11][Rule12]

[규칙 13]

enumeration 과 pattern을 제외한 나머지 Facet는 클래스의 속성으로 표기한다.

(스키마 소스)

```
<simpleType name="Tel">
  <restriction base=xsd:integer">
    <minlength value="6"/>
    <maxlength value="12"/>
  </restriction>
</simpleType>
```



[그림 11] [규칙 13]의 스키마 소스와 UML 표기

[Fig. 11] UML notation and Schema source of [Rule 13]

4.4 기타 사상 규칙

앞에서 규칙으로 규정하지 않은 namespace, extension등은 다음 <표 2>의 XML 스키마와 UML 프로파일에 있는 방식대로 표기한다.

<표 2> XML 스키마와 UML 프로파일

<Table 2> XML Schem and UML Profile

XML 스키마	UML 구문	의미
Schema namespace	Package Package streotype = w3cSchema	모든 스키마를 나타냄 패키지 다이어그램에서 <<w3c>>스테레오타입 선언
extension	일반화	일반화 관계

5. 실험 및 고찰

5.1 실험

사상 규칙을 구현하는 알고리즘을 구현하였다. XML 스키마와 문서 인스턴스를 입력받아 XML 스키마 다이어그램을 생성하는 알고리즘은 다음과 같다.

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

begin {
switch (선언된 스키마 엘리먼트) {
case : element {
  if ( 속성을 정의하는 엔티티선언) then
    entity_change() // 엔티티 교체
  else {
    create_class()  // 클래스생성
  while ( content_empty() ) {
    // 컨텐트 모델이sequence,choice,all인경우
    if ( sequence or choice or all ) then {
      create_attribute() //속성추가
      create_aggregation() //집단화 생성
```

```
    create_subclass() //하위클래스 생성
   }
   else  // 'Datatype'인경우
    create_inherit()  //기본타입 상속
}}}
case : attribute or facet {
 //엘리먼트에 속성 및 타입 추가
 attr_Create_function()
 if ( attribute_name == 'enumeration' ) then
{
   create_class()
   create_dependency() // 의존관계
   //<<enumeration>> 스테레오타입
   create_enumeration_stereotype()   }
 elseif ( attribute_name == 'pattern')  then
{
   create_class()
   create_dependency()
   //<<pattern>> 스테레오타입
   create_pattern_stereotype()    }
 else {
   create_class()
   create_dependency()
   //클래스의 속성으로 표기
   depand_attribute()   }}}
}
end;
```

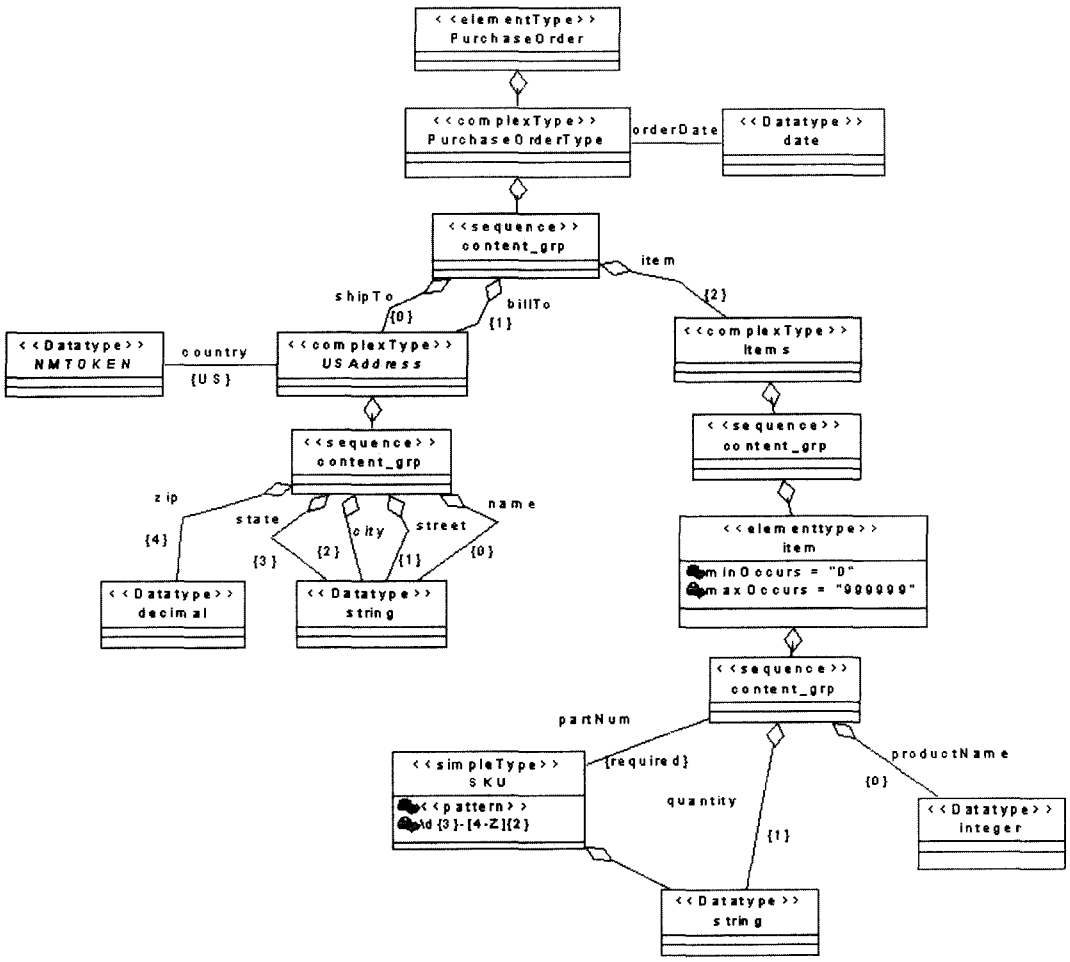
다음의 예는 간단한 구매주문서 문서의 데이터 모델이다. XML 스키마로 표현된 문서구조를 사상을 통해 UML로 표기한다. 다음의 XML 스키마 정의는 XML Purchase Order(구매주문서) 인스턴스을 위해 사용된 XML 문서형으로 [그림 12]와 같이 기술하였다.

```
<xsd:schema  xmlns:xsd="http://www.w3.org/200
1/XMLSchema">
 <xsd:annotation>
  <xsd:documentation xml:lang="en">
   Purchase order schema for Example.com.
   Copyright 2000 Example.com. All rights res
erved.
  </xsd:documentation>
 </xsd:annotation>

 <xsd:element name="purchaseOrder" type="Purc
haseOrderType"/>

  <xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
   <xsd:element name="shipTo" type="USAddress"/>
   <xsd:element name="billTo" type="USAddress"/>
   <xsd:element name="items"  type="Items"/>
  </xsd:sequence>
  <xsd:attribute name="orderDate" type="xsd:
date"/>
 </xsd:complexType>

 <xsd:complexType name="USAddress">
  <xsd:sequence>
   <xsd:element name="name"   type="xsd:string"/>
   <xsd:element name="street" type="xsd:string"/>
   <xsd:element name="city"   type="xsd:string"/>
   <xsd:element name="state"  type="xsd:string"/>
   <xsd:element name="zip"    type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN"
    fixed="US"/>
 </xsd:complexType>

 <xsd:complexType name="Items">
  <xsd:sequence>
   <xsd:element name="item" minOccurs="0" max
Occurs="unbounded">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="productName" type="x
sd:string"/>
      <xsd:element name="quantity" type="xsd:
integer"/>
     </xsd:sequence>
     <xsd:attribute name="partNum" type="SKU"
use="required"/>
    </xsd:complexType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>

 <!-- Stock Keeping Unit, a code for identify
ing products -->
 <xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
   <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
 </xsd:simpleType>

</xsd:schema>
```

[그림 12] Purchase Order의 XML 스키마

[Fig. 12] XML Schema of Purchase Order

[그림 12]의 XML 스키마를 본 연구에서 제안한 사상 규칙을 적용시켜 알고리즘으로 구현한 결과는 [그림 13]이다. [그림 13]은 UML 스키마 다이어그램 형태의 XML 문서 구조 다이어그램이다.



[그림 13] Purchase Order의 UML 스키마 다이어그램

[Fig. 13] UML S초등 Diagram of Purchase Order

5.2 고찰

본 논문은 XML 스키마를 객체지향 모델링 하기 위하여 UML의 확장 메카니즘인 스테레오타입 다이어그램을 이용한 XML 스키마 다이어그램을 제안에 관한 연구이다. 본 연구에서 제안한 [그림 13]의 UML 스키마 다이어그램과 [1]을 비교해보면 두 다이어그램의 표기법의 차이는 <표 3>과 같다. <표 3>에서 살펴보면 사상 표준이 틀리기 때문에 정확히 비교가 되지 않지만 Facet의 종류인 enumeration 과 pattern을 따로 스테레오타입으로 표기하지 않고 클래스의 속성에서 스테레오타입으로 분류하였다. 또한 그룹 표기를 그림에서 쉽게 알아볼 수 있도록 DTD 형태인 그룹 스테레오타입으로 표기하였으며, W3C에서 발표한 XML 스키마의 최종 권고안(Recommendation)을 바탕으로 사상 규칙을 정의하였다.

본 사상 명세의 특징 및 장점은 그룹 표기를 그룹 스테레오 타입으로 분류하여 가독성 및 이해성을 높였으며, 3가지 데이터형을 적절히 사용하여 모듈화 함으로서 재사용 가능하고 유연성이 좋은 스키마를 만들 수 있도록 하였다.

미흡한 점으로는 그룹 표기를 그룹 스테레오 타입으로 분류하여 가독성 및 이해성을 높였으나 오버헤드가 발생하였다.

<표 3> 타 UML 스키마 다이어그램과의 표기법 비교
<Table 3> Compare to other UML Schema Diagram

		[1]	본 논문
XML 스키마	사상 표준	SOX1.1[15]	W3C XML Schema 1.0 권고안[2,3,4]
	패키지	<<sox>>	<<Schema>>
	모델 그룹	<<sequence>>,<<choice>>	<<sequence>>,<<choice>>,<<all>>
	데이터형	<<enumeration>>,<<scalar>>,<<varchar>>	<<Datatype>>,<<simpleType>>,<<complexType>>
	속성	<<implied>>,<<required>>,<<default>>,<<fixed>>	<<implied>>,<<required>>,<<default>>,<<fixed>>,<<optional>>,<<prohibited>> Facet중<<enumeration>>,<<pattern>>만 구분지어 표기
장점		그룹 표기를 그룹 스테레오 타입으로 분류하여 가독성 및 이해성을 높임 변환구문이 단순하여 변환이 용이	3가지 데이터형을 적절히 사용하여 모듈화 재사용 가능하고 유연성이 좋은 스키마 생성 그룹 표기를 그룹 스테레오 타입으로 분류하여 가독성 및 이해성을 높임
단점		그룹 표기에서 오버헤드 발생 속성표기가 단순	그룹 표기를 그룹 스테레오 타입으로 분류하여 오버헤드 발생

6. 결론 및 향후 연구 과제

본 연구에서는 XML 스키마를 객체지향 모델링하기 위해 UML 스테레오타입 다이어그램을 이용한 XML 스키마 다이어그램을 제안하였다. XML DTD와 XML 스키마의 차이점을 파악하여 XML DTD에서 XML 스키마로 사상시킨 내용과 XML 스키마 권고안의 내용을 추가 확장하여 제안하였다. 또한 이런 내용을 바탕으로 변환 알고리즘을 구현하였다. 그러나 XML 문서에서 다른 자원(URL 등)을 가리키는 링크 구조를 포함하는 경우에는 본 연구에서 제한하였다. 그래서 단순 링크를 나타내는 'HREF'나 확장 링크인 'LOCATOR'는 생략하였다.

본 연구의 XML 스키마 다이어그램은 복잡한 XML 스키마를 시각화하여 쉽게 파악할 수 있도록 하였으며, 또한 XML 문서 개발자들이 UML 방법을 활용해 설계하면 재사용성이 높고 유연성이 뛰어난 문서 구조를 만들 수가 있다. 이러한 방법으로 XML 스키마를 효율적으로 모델링하면 저장, 변환, 전달 과정의 생명 주기가 효율적으로 관리할 수 있으며, 업무의 변화에 따라 재사용 될 수 있는 문서 구조가 된다. 또한 XML 문서의 문법을 파악할 필요 없이 UML 클래스 다이어그램 형태의 모델링을 데이터베이스 스키마로 변환할 수가 있으므로 데이터베이스 저장이 용이해진다.

향후 연구 과제는 모델링 분야인 본 연구의 UML 스키마 다이어그램으로 XML 스키마를 생성하는 변환기를 개발하고, XML 스키마로 표현한 XML 기반의 컴포넌트를 검색하는 에이전트에 대한 연구를 하는 것이다.

※ 참고 문헌

[1] Grady Booch et al., "UML for XML Schem a Mapping Specification", http://www.rationa l.com/media/uml/resources/media/uml_xmlsche ma33.pdf, 1999.

[2] W3C, "XML Schema Part 0: Primer", http:// www.w3.org/TR/xmlschema-0, 2 May 2001.

[3] W3C, "XML Schema Part 1:Structures", htt p://www.w3.org/TR/xmlschema-1, 2 May 2 001.

[4] W3C, "XML Schema Part 2:Datatypes", htt p://www.w3.org/TR/xmlschema-2, 2 May 20 01.

[5] Eric van der Vlist, "Using W3C XML Sche ma", http://www.xml.com, 2000.

[6] Norman Walsh, "Understanding XML Schem as", http://www.xml.com, 2000.

[7] 김채미, 최학열, 김심석, "전문가와 함께하는 XML 캠프", 마이트프레스, 2001.

[8] Roger L. Costello, "XML Schema: Best Pra ctice" http://www.xfront.com/BestPracticesHo mepage.html, 22 April 2001.

[9] Grady Booch, James Rumbaugh, Ivar Jacobs on, "The Unified Modeling Language User Guide", Addison-Wesley, 1999.

[10] James Rumbaugh, Ivar Jacobson, Grady Boo ch, "The Unified Modeling Language Refere nce Manual", Addison-Wesley, 1999.

[11] Eric van Herwijnen, "Practical SGML", Klu wer Academic Publishers, 1994.

[12] 박인호외, "XOMT : SGML DTD 설계를 위 한 객체 다이아그래밍 기법", 정보과학회논 문지(C), 제3권, 제3호, pp.228-237, 1997.6.

[13] 채원석, 하얀, 김용성, "UML 클래스 다이어 그램을 이용한 XML 문서 구조 다이어그래 밍", 정보처리학회논문지, 제6권, 제10호, pp. 508-520, 1999.10.

[14] Rational Software White Paper, "Migrating f rom XML DTD to XML-Schema using UM L", http://www.rational.com/product/whitepape rs, 2000.

[15] Matthew Fuchs et. A1, "Schema for Object-oriented XML", W3C, 1998, http://www.w3. org/Sumission/1998/15.

[16] Jim Conallen, "Working with XML Docume nts in UML", ROSE Architect Magazine S pring Issue, April 2000.

[17] Robin Cover, "The XML Cover Pages-XML Schemas", http://www.oasis-open.org/cover/sc hemas.html, 26 April 2001.

[18] Joseph Schmuller, "Teach Yourself UML ", SAMS, 1999.

조 정 길

1987년 숭실대학교 정보과학대학 졸업
1993년 숭실대학교 정보과학대학 석사(이학석사)
1996년~현재 남서울대학교 컴퓨터학과 겸임교수
2000년~현재 충북대학교 전자계산학과 박사과정 수료
관심분야 : 객체지향 자료 모델링, XML 문서관리, 질의처리, 정보검색