# 메타데이터를 이용한 XML 스키마언어

최한용*, 이돈양**
*한북대학교 컴퓨터공학과
**경인여자대학 전산정보학과
e-mail:hychoi@hanbuk.ac.kr*, dylee62@empal.com**

# Metadata Using XML Schema-Language

Han-Yong Choi*, Don-Yang Lee**
*Dept of Computer Engineering, Han-Buk University
**Dept of Computer Engineering, Kyung-In Woman's College

요 약

소프트웨어 설계와 관련하여 OMG의 UML은 객체지향모델링에 대해서 표준화된 언어에 지원이 가능하여 널리 사용되고 있다. 그리고 마크업언어로는 일반적으로 DTD와 XML 스키마를 많이 사용하고 있다. 본 연구에서는 클래스내의 단위 엘리먼트의 속성을 부여할 수 있고 모델내의 클래스의 관계를 표현할 수 있는 수퍼클래스와 서브클래스에 대한 정확한 타입의 속성을 표현할 수 있도록 하였다. 그리고 엘리먼트에 대한 어트리뷰트를 표현하는데 다양하고 세부적인 데이터타입이 지원되도록 하여 XMI 메타모델 기반의 메타데이타 생성이 가능한 도구를 설계/구현 하였다.

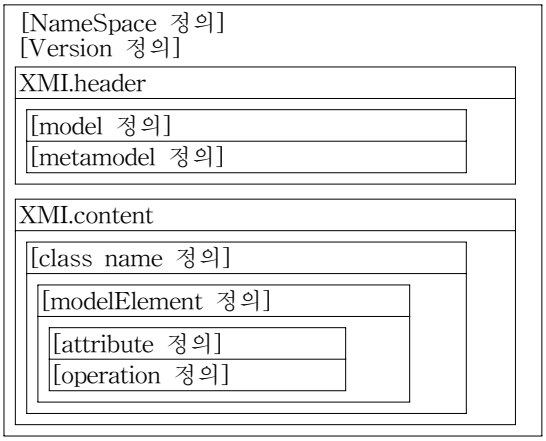
## 1. 서론

일반적으로 소프트웨어설계에서는 기술적인 실현 가능성과 정확성, 그리고 사용자의 요구사항들을 모두 포함한다[1]. 소프트웨어 설계와 관련하여 OMG의 UML은 객체지향모델링에 대해서 표준화된 언어에 지원이 가능하여 널리 사용되고 있다[2]. 그리고 마크업언어로는 일반적으로 DTD[3]와 XML 스키마[4]를 많이 사용하고 있다. DTD는 현재 널리 사용되고 있으며 광범위한 도구의 지원을 받고 있고, XML 스키마는 트리 구조의 문법을 사용하여 Document와 다양한 데이터 타입을 표현할 수 있다. 그리고 XMI 메타모델로 정의된 세부적인 클래스의 메타데이터에 대한 생성방법정의, 생성도구설계 및 구현에 중심을 두었다. 현재 사용되고 있는 메타데이터설계 및 생성도구로 XML SPY[5], PIXEE[6], 그리고 다산의 XML Builder[7] 등이 있다. 그러나 일반적인 XML Document에서 요구되어지는 것들에 중점을 두고 있으며 세부적인 스키마를 생성하지 못하고 있다. 본 논문에서는 클래스내의 단위 엘리먼트의 속성을 부여할 수 있고 모델내의 클래스의 관계를 표현할 수 있는 스키마언어 생성도구를 설계/구현 하였다. 그리고 엘리먼트에 대한 어트리뷰트를 표현하는데 다양하고 세부적인 데이터타입이 지원되도록 하여 XMI 메타모델 기반의 메타데이타 생성이 가능하도록 하였다.

## 2. XMI 메타데이터 저장소

### 2.1 XMI 메타모델 정의

일반적으로 객체지향모델링(object oriented modeling)에서는 UML을 이용한 방법이 많이 사용되고 있으나[8], 본 논문에서는 그림 1과 같이 UML에 의해 산출된 각 클래스 다이어그램의 형식들을 XML형식으로 자동 변환시키는 것을 표준화시키기 위한 방법으로 제안된 XMI(XML Metadata Interchange)를 이용하였다. 따라서 UML로 작성된 각종 다이어그램들은 XMI의 메타모델로 작성될 수 있으며, 또한 규정에 따라 XML로 표현될 수 있도록 하였다. 그리고 메타모델의 저장소 설계에서는 관계형 데이터베이스(relational database)를 이용하여 메타데이터의 무결성을 위한 정규화가 적용된 테이블을 작성하였다.




(그림 1) 메타모델 정의

### 2.2 XMI 관계형 데이터베이스 모델

그림 2와 같이 관계형 데이터베이스로 XMI를 표현하기 위해서 데이터베이스 이름, 테이블, 컬럼 등을 tag를 생성하였다. 또한, 관계형 데이터베이스와 테이블들은 각각의 종속된 테이블과 컬럼을 가질 수 있도록 하였다. 이는 UML로 정의된 디자인패턴을 XMI를 이용하여 표준화시켜 어플리케이션을 설계하는데 소프트웨어지원, 저장소 및 데이터베이스 스키마에 대한 개방된 상호교류가 가능하도록 한 것이다. 그리고 <Column.name>을 이용하여 데이터베이스내의 엘리먼트나 어트리뷰트, 오퍼레이션에 대한 메타데이터를 작성하였다.

```
<RelationalDatabase>
   <RelationalDatabase.name>
     데이터베이스 이름
   </RelationalDatabase>
   <RelationalDatabase.tables>
      <Table>
         <Table.name>
           데이블 이름
         </Table.name>
            <Table.columns>
                <Column>
                    <Column.name>
                      칼럼 이름
                    </Column.name>
                </Column>
            </Table.columns>
         </Table.name>
      </Table>
    </RelationalDatabase.tables>
</RelationalDatabase>
```

(그림 2) XMI 관계형 데이터베이스 모델

### 2.3 XML 스키마언어 데이터타입 분류

앞에서 정의한 클래스의 XML 메타데이터의 마크업 언어 정의에 사용되는 컴포넌트의 형식 중 본 논문에서는 심플타입(simple type)과 콤플렉스 타입(complex type)으로 분류하였다. XML 스키마에서 데이터타입은 그림 3과 같이 이미 정의가 되어 사용되고 있는 내장형 심플타입(built-in simple type)과 사용자가 정의하여 사용하는 사용자 심플타입(user define simple type) 그리고 사용자 정의 콤플렉스 타입(user define complex type)으로 정의하였다. 또한 데이터가 정의되는 위치에 따라 글로벌 데이터 타입(global data type)과 로컬 데이터 타입(local data type)으로 분류할 수 있다.

```
<simpleType  name = "simple type name">
   (restriction | list | union)
</simpleType>
```

```
<simpleType>
   (restriction | list | union)
</simpleType>
```

(그림 3) 글로벌 심플타입, 로컬 심플타입 정의

심플타입에서 자식 엘리먼트로 올 수 있는 것은 "restriction", "list", "union" 가 있으며 이 중하나를 선택하여 기술할 수 있다. “restriction"은 내장된 심플타입 또는 이미 정의되어 사용되는 사용자정의 심플타입을 제한하여 새로운 심플타입을 정의할 때 사용하고, "list"는 공백 문자열로 분리된 토큰(token)들의 리스트를 값으로 갖고자하는 타입을 정의할 때 사용한다. 그리고 "union"은 여러 개의 심플타입을 결합하여 여러 종류의 데이터 값을 갖는 경우 사용한다. 또 다른 타입으로 사용되는 콤플렉스 타입은 속성을 가지거나 자식 엘리먼트를 가지는 엘리먼트의 선언에 필요한 타입으로써 이 역시 글로벌 콤플렉스 타입과 로컬 콤플렉스 타입이 있다. 그리고 그림 4와 같이 <sequence>를 사용한 자식 엘리먼트의 사용에서는 “순차적 자식 엘리먼트 콤플렉스 타입”과 “선택적 자식 엘리먼트 콤플렉스 타입” 정의를 사용할 수 있다.

```
<complexType name="complex type name">
  <sequence>
    Element ...
  </sequence>
</complexType>
```

(그림 4) 순차적 자식 엘리먼트 콤플렉스 타입

<sequence>의 자식 엘리먼트로 대개 여러개의 엘리먼트가 오지만 한 개의 엘리먼트만 사용되는 경우에도 반드시 <sequence> 엘리먼트를 삽입해야 한다. 그리고 그림 5와 같이 “선택적 자식 엘리먼트

콤플렉스 타입" 에서는 <choice>를 사용하여 선택적으로 엘리먼트를 사용할 수 있다.

```
<complexType name="complex type name">
  <sequence>
    Elements ...
    <choice minOccurs="minimum count"
maxOccurs="maximum count"
      Elements ...
    </choice>
    Elements ...
  </sequence>
</complexType>
```

(그림 5) 선택적 자식 엘리먼트 콤플렉스 타입

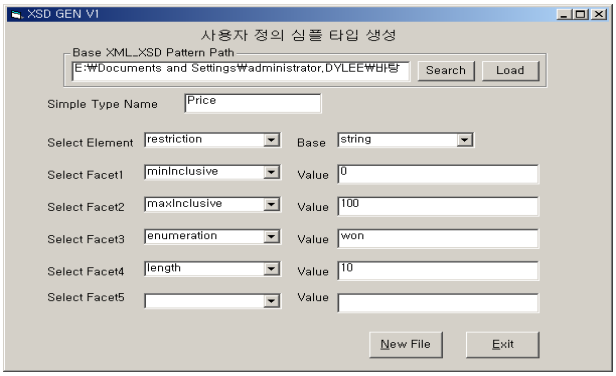
### 3. XML 스키마언어 생성

그림 6은 사용자 정의 심플타입 XML 스키마언어를 생성하기 위한 알고리즘이다. 스키마언어의 생성은 XML 문서의 루트 노드 오브젝트를 설정한 후, 루트 엘리먼트 객체를 설정하였다. 그리고 심플타입 엘리먼트 name 속성 생성과 엘리먼트의 종류를 선택하도록 하였으며, 마지막부분에서는 엘리먼트의 facet를 선택할 수 있도록 하였다.

```
begin {simpleType}
  Level1 <- Root    '
  Level2 <- simpleType   '
  if simpleType(name) != " "
    attribute(name) <- name
  end if
  switch simpleTypeElement
        case 0
            Level3 <- restriction
        case 1
            Level3 <- list
        case 2
            Level3 <- union
  end switch
  if simpleTypeElement(base) != " "
    switch simpleTypeElementBase
        case 0
            base <- string
        case 1
            base <- boolean
        .....
        case 18
            base <- byte
    end switch
  end if
  Level4 <- facet
  switch facet
        case 0
            value <- minExclusive
        case 1
            value <- minInclusive
        .....
        case 10
            value <- pattern
  end switch
end {simpleType}
```

(그림 6) 심플타입 XML 스키마언어 생성 알고리즘

그림 7은 심플타입 XML 스키마언어 생성방법에 대한 프로그램정의를 이용하여 구현한 도구이다. 마크업언어를 생성하기위한 방법으로 DOM 트리의 기본 파일형식을 읽어 들여 파싱 하였으며, 새로운 사용자 정의 심플타입 스키마언어를 생성하기 위한 이름과 엘리먼트 뿐만 아니라 패싯을 선택할 수 있도록 구성하였다.



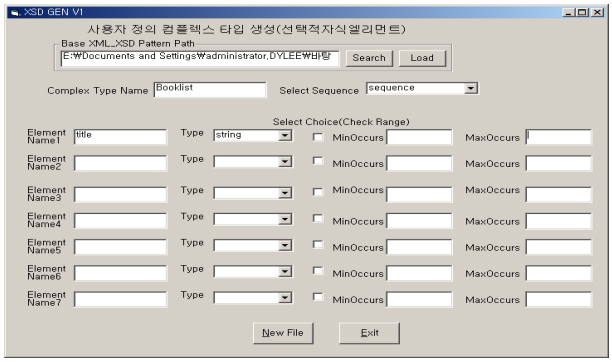

(그림 7) 사용자 정의 심플타입 XML 스키마 도구

그리고 사용자 정의 콤플렉스타입 XML 스키마언어를 생성하기 위한 방법으로 그림 8의 알고리즘에서는 요구되는 각 엘리먼트들을 단계적으로 생성할 수 있도록 하였다. 기본적인 생성방법은 그림 6과 같으며, sequence 엘리먼트를 선택하는 경우에 따라서 생성되는 자식트리의 형태를 알 수 있다.

```
begin {complexType}
  Level1 <- RootType
  Level2 <- complexType
  if complexType(name) != " "
    attribute(name) <- name
  end if
  if sequence != " "
    Level3 <- sequence
  end if
    Level4 <- element
  switch ElementBase
        case 0
            base <- string
        case 1
            base <- boolean
        .....
        case 18
            base <- byte
  end switch
end {complexType}
```

(그림 8) 사용자정의 콤플렉스타입(선택적자식엘리먼트) XML 스키마생성 알고리즘

그림 9의 사용자정의 콤플렉스타입(선택적자식엘리먼트) XML 스키마언어 도구는 그림 8의 프로그램의 작성알고리즘에 의해 생성된 것이다. 여기서는 엘리먼트의 속성으로 DTD에서 지원하지 못하는 광범위한 데이터 타입을 정의하게 하고, 최소발생횟수와 최대발생횟수에 대한 속성을 추가하였다.




(그림 9) 콤플렉스타입 XML 스키마 도구

## 4. 평가 및 결론

표 1과 같이 PIXEE는 DTD의 기본구조와 매우 흡사하고, SAX 기반의 파서형태를 이용한다. 이는 간단한 XML 문서나 어플리케이션에서는 복잡하기 않아 사용하기가 쉽지만 다양한 데이터타입을 정의하거나 서로 다른 타입을 변환하고자 할 때는 매우 어려울 뿐만 아니라 다양성을 갖지 못하고 있다. 그리고 DTD 기반으로 마크업언어를 생성하므로 클래스 엘리먼트에 대한 세부적인 타입으로 분류하기가 어렵다. XML SPY는 다양한 데이터타입의 지원에 대해서 심플타입과 콤플렉스타입을 이용한 스키마 생성이 가능하지만 메타데이터 마크업언어의 생성에는 적절하지 못한 일반적인 XML 메타데이터 기반의 마크업언어 생성에 기반을 두고 있다. 그리고 클래스내의 단위 엘리먼트의 속성을 부여할 수 있는 방법이 없기 때문에 모델내의 클래스의 관계를 표현할 수 있는 수퍼클래스와 서브클래스에 대한 적절한 타입의 속성을 표현할 수 없다.

| 기준 \ 시스템 | | PIXEE | XML SPY | 본 논문제안 |
|---|---|---|---|---|
| 마크업언어 | | DTD | DTD, XML 스키마 | XML 스키마 |
| 클래스 엘리먼트생성 | Simple Type | 제한적 | 가능 | 가능 |
| | Complex Type | 제한적 | 가능 | 가능 |
| 영역별 엘리먼트 생성 | | 제한적 | 제한적 | 가능 |
| 문서작성형식 | | 트리기반 | 트리기반 | 텍스트기반 |
| 데이터 타입 | 어트리뷰트 | 제한적적용 | 세부적적용 | 세부적적용 |
| | 엘리먼트 | - | - | 세부적적용 |
| 비 고 | | 일반 메타모델 기반 | 일반 메타모델 기반 | XMI 메타모델 기반 |

표 1. 제한적 메타모델적용기반 시스템비교

본 논문에서는 XMI 메타모델 기반의 메타데이터 마크업언어 생성에 대한 엘리먼트와 어트리뷰트에 대한 세부적인 데이터의 표현이 가능할 뿐만 아니라 abstract 클래스와 concrete 클래스의 관계에 대한 속성도 표현이 가능하도록 하기 위해서 텍스트 형태의 마크업언어를 생성할 수 있다.

마지막으로 본 연구에서는 메타모델과 메타데이터에 대한 부분적인 연구만 이루어지고 있어, 앞으로는 도식화된 패턴이나 클래스 다이어그램에 대한 메타모델 및 메타데이터, 관계형 데이터베이스 작성, 마크업언어 생성 등 이와 관련된 것들에 대한 연계적인 자동화 도구 개발이 요구되어 진다.

## 참고문헌

[1] Gregor Engels, Luuk Groenewegen. "Object-Oriented Modeling: A Roadmap" In proceedings of "The Future of Software Engineering 2000", Editor: Anthony Finkelstein, International Conference on Software Engineering.

[2] "OMG Unified Modeling Language Specification (draft)" Version 1.3. beta R7, June 1999.

[3] Tim Bray, Jean Paoli, and C.M. Sperberg-McQueen, editors. Extensible Markup Language(XML) 1.0. World Wide Web Consortium, 1998.

[4] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn, editors. XML Schema Part 1: Structures. World Wide Web Consortium, 2000.

[5] xmlspy Enterprise Edition User and Reference Manual, www.xmlspy.com/document/xmlspy2004.pdf. 2004

[6] Robert Kosara, Klaus Hammermuller, and Silvia Miksch. Codesigning XML-bsaed language and classes with pontifex. Technical Report Asgaard-TR-2000-1, Vienna University of Technology, Institute of Software Technology, Vienna, Austria 2000.

[7] http://www.tagfree.com

[8] Wu, I. C.; Hsieh, S. H, "An UML-XML-RDB Model Mapping Solution for Facilitating Information Standardization and Sharing in Construction Industry", International Symposium on Automation and Robotics in Construction, 19th (ISARC). Proceedings. National Institute of Standards and Technology, Gaithersburg, Maryland. September 23-25, 2002, 317-321 pp, 2002