XMI기반 클래스의 메타데이터생성

Generation of Class MetaData Based on XMI

이상식*, 최한용**
송호대학 보건의료기기과*, 한북대학교 컴퓨터정보학과**

Sang-Sik Lee(leess@songho.ac.kr)*, Han-Yong Choi(hychoi@hanbuk.ac.kr)**

요약

XMI 메타모델과 XML 메타데이터를 이용한 클래스에 대한 연구는 일반적으로 이용되고 있는 XML 메타데이터의 생성과 상당한 차이점이 있다. 대부분의 XML 시스템은 에디터기능과 데이터베이스 연동, 등 마크업언어의 생성부분에 많은 비중을 두고 개발하고 있다. 그러나 본 연구는 이와 달리 XMI 메타모델에서 추출되는 클래스 메타데이터의 마크업언어를 생성하는데 중점을 두었다. 또한 클래스내의 단위 엘리먼트의 속성부여와 모델내의 클래스 관계를 표현할 수 있도록 하였다. 마크업언어의 생성에서는 XML 스키마를 이용하여 세부적인 데이터타입의 선언이 가능하도록 하고 있다.

■ 중심어 : | 모델링언어 | 메타데이터 교환표준 | 문서 형식 정의 | 확장성 생성언어 스키마 | 메타데이터 |

Abstract

Study on the class using XMI Meta model and XML MetaDats has significant difference from the method of Data creation which is widely used. Most of MXL System are focusing on the editor funcition, Database connection and Generation of Markup language. Unlikelly, however, this study has focused on the creation of Markup language of Class MetaData which are extracted from MXI data modedl. In addition to that, the attribute of unit element within the class and the relationship between the classes within the model were set to be given and expressed respectively. For the generation of Markup language, XML schema was used to declare the detail data type.

■ keyword : | UML(Unified Modeling Language) | XMI(XML Metadata Interchange Format) | DTD(Document Type Definition) | XML(Extensible Markup Language) Schema | Metadata |

Ⅰ. 서론

일반적으로 소프트웨어설계에서는 기술적인 실현 가능성과 정확성, 그리고 사용자의 요구사항들을 모두 포함한다[1]. 소프트웨어 설계와 관련하여 최근에는 객체지향모델링[2]을 이용한 방법이 많이 사용되고 있으며, 이는 사용자들의 요구사항에 대한 관점에서 최신의 기술을 적용하고 있다. 특히, OMG의 UML은 객체지향모델링에 대해서 표준화된 언어에 지원이 가능하여 널리 사용되고 있다[3]. 본 연구에서는 UML을 이용하여 정의된 클래스 모듈을 XMI의 메타모델로 정의하고 다시 여기서 메터데이터를 추출하는 연구를 하였다.

* 본 연구는 한국학술진흥재단 연구과제(과제번호 : 20090291000)로 수행되었습니다.

접수번호 : #090813-002
접수일자 : 2009년 08월 13일
심사완료일 : 2009년 09월 10일
교신저자 : 이상식, e-mail : leess@songho.ac.kr

그리고 XMI 메타모델에 대한 이름, 테이블, 컬럼 등의 tag를 생성하여 관계형 데이터베이스를 구축하여, 이를 기반으로 한 단위클래스의 테이블을 만들어 XML 메타데이터를 생성할 수 있도록 하였다. 특히 중점을 둔 부분은 메타모델에서 클래스를 메타데이터로 생성하는 부분이다.

XML에서 데이터를 이용한 어플리케이션 개발에서는 두 가지의 기능에 대해서 언급할 수 있는데, 하나는 마크업언어 설계에 관한 것이고, 다른 하나는 적절한 클래스의 생성에 대한 것이다. 마크업언어로는 일반적으로 DTD[4]와 XML 스키마[5]를 많이 사용하고 있다. DTD는 현재 널리 사용되고 있으며 광범위한 툴(tools)의 지원을 받고 있고, XML 스키마는 트리 구조의 문법을 사용하여 Document와 다양한 데이터 타입을 표현할 수 있다. 본 연구에서는 XML 스키마를 가지고 연구하였다. 그리고 XMI 메타모델로 정의된 세부적인 클래스의 메타데이터에 대한 생성방법정의, 생성도구설계 및 구현에 중심을 두었다. 현재 사용되고 있는 메타데이터설계 및 생성도구로 XML SPY[6], PIXEE[7], 그리고 다산의 XML Builder[8] 등이 있으나 이들 대부분이 일반적인 XML Document에서 요구되어지는 것들에 중점을 두고 있어 클래스에 대한 데이터타입(data type), 어트리뷰트(attribute), 엘리먼트(element), 인허리턴스(inheritance) 등의 세부적인 정의에 대해서는 어려움을 가지고 있다. 이를 보완하기 위한 방법으로 본 연구에서는 클래스내의 단위 엘리먼트의 속성을 부여할 수 있고 모델내의 클래스의 관계를 표현할 수 있는 SuperClass와 SubClass에 대한 적절한 타입의 속성을 표현할 수 있도록 하였다. 그리고 엘리먼트에 대한 어트리뷰트를 표현하는데 다양하고 세부적인 데이터타입이 지원되도록 하여 XMI 메타모델 기반의 메타데이타 생성이 가능하도록 하였다.

II. 관련연구

2.1 XMI 기반 패턴모델

UML(Unified Modeling Language)이 소프트웨어 모델링에 대한 표준으로 빠르게 받아들어지게 되는 것과 더불어 UML 모델의 기능적인 변환에 대한 기술적인 중요성이 증가되고 있다. UML 설계에서는 시스템의 개발에 사용되는 언어뿐만 아니라 이에 종속된 세부적인 것들의 차이들로 인하여 모델의 변환에 대한 서로 다른 표현의 개념들을 가질 수 있다[9][10].

본 논문에서는 이런 문제점들을 해결하기 위한 방법으로 OMG의 XMI(XML Metadata Interchange Format)[11]를 이용한 방법을 도입하고자 하였다. XMI는 서로 다른 환경에 분산되어 저장되어 있는 메타데이터와 UML 모델링에서 사용되고 있는 데이터나 메타데이터와의 원활한 변환 및 교환을 목적으로 표준화된 모델링 방법이다.

XMI 엘리먼트는 XMI.header, XMI.content, XMI.extensions, XMI.documentation, XMI.owner, XMI.contact, XMI.longDescription,, XMI.shortDescription, XMI.exporter, XMI.exporterVersion, XMI.expor terID, XMI.notice, XMI.model, XMI.metamodel, XMI. metametamodel, XMI.import, XMI.difference, XMI.d elete, XMI.add, XMI.replace, XMI.reference으로 구성이 되어있다. 그러나 본 논문에서는 UML의 클래스 모델에 대한 XMI 메타모델로의 변환에는 Top Level에 해당하는 <XMI.header>, <XMI.content>, <XMI.difference>, <XMI.extensions>의 4개의 엘리먼트 만을 포함시켜서 모델을 구성하였다.

본 연구는 서론에 이어 2장에서는 관련연구를 기술하였고 3장에서는 XMI 메타데이터 저장소의 설계와 메타모델 생성에 관하여 기술하였다. 4장에서는 XML 기반 클래스 설계 및 생성, 5장에서는 XML 스키마언어 생성도구 설계 및 구현에 관해 기술하고 끝으로 6장에서 결론 및 평가를 제시하였다.

2.2 XMI 데이터베이스 저장소

기존의 일반적인 방법에서 CASE 도구를 이용하여 데이터베이스 저장소를 구축하는 방법으로는 UML의 사용자 모델링정보를 모두 저장하였다[9][12]. 이는 정보저장소 내의 컴포넌트나 패턴의 정보가 정형화 되어 있지 않고, 단일 구조로서 모든 컴포넌트나 패턴의 객

체가 독립적인 영역에 저장되었다. 이의 방안으로 모델링 설계 시 중복성을 줄이기 위해 설계상의 정보를 공유 저장소 영역과 개인 저장소 영역으로 분리한 혼합형 구조를 설계하였다[13]. 이것은 다수의 클래스로 구성된 컴포넌트나 패턴 기반으로 저장되었다. 이로 인해 조립 시 점차적으로 합성되는 패턴에 대한 클래스가 증가되므로 어트리뷰트 및 오퍼레이션이 증가되어 시스템의 크기 및 코드라인의 증가에 대한 문제점들이 발생되었다. 또한 클래스 합성에서는 클래스 위주가 아닌 컴포넌트나 패턴위주로 합성이 이루어져 공통의 클래스를 추출하기가 어려웠다. 그리고 [14]에서는 XMI 기반으로 표준의 데이터 모델을 생성하고 메타데이터로 분리하여 DB 저장소에 저장하는 방법을 소개하였다.

Ⅲ. XMI 메타데이터 저장소

3.1 XMI 메타데이터 저장소 설계

본 연구에서는 XMI 메타모델의 4개의 엘리먼트를 저장소에 저장하기 위해서 <XMI.header>, <XMI.content>, <XMI.difference>, <XMI.extensions>를 각각의 클래스 부품으로 구분하여 추출하였다. 여기서 클래스를 SuperClass와 SubClass로 분리하여 이들 사이의 상속관계에 대한 정보를 기술하고자 하였다.

메타데이터 저장소 설계에서 기존에는 Entity Relationship Diagram에 대한 XML DTD 위주의 DB 알고리즘을 설계와 UML Class Diagram의 XML DTD의 변환목적에 중점을 두고 있다.[15][16][17][18]

그러나 본 논문에서는 DTD 보다는 XML 스키마 기반에 중점을 두어 연구하였다.

[그림 1] 의 Use Case 다이어그램은 초기 시스템 개발 시 사용자와 개발자가 충분히 요구사항을 반영하고 변경할 수 있는 대화 수단이며, 시스템 사용자 혹은 외부시스템을 나타내는 Actor들과 관련하여 수행되는 행위인 Use Case의 상호작용으로 표현된다. 본 시스템에서는 Use Case 다이어그램 사용자인 사용자와 관리자, 메타모델 DB 등 세 개의 Actor로 구성하였다. 그리고 사용자 로그인과 XMI_header, XMI_ contents, XMI_difference, XMI_extensions, Meta Model Management 등 6개의 Use Case를 사용하여 상호간의 관계파악이 가능해 메타모델DB를 설계하는데도 참조할 수 있다.

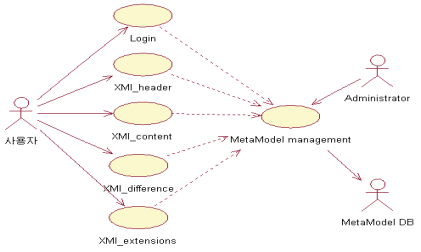

그림 1. XMI 메타모델 Use Case 다이어그램

[그림 2] 의 Sequence Diagram은 시스템설계자가 XMI_header에 해당하는 abstract 클래스와 나머지 세 개의 concrete 클래스에 대한 데이터베이스 생성의 모델링을 실제적인 단계로 보여주고 있다. 첫 번째 사용자로 하여금 DB Open에 대한 Logon 권한부여와 DB 구축 후 모델내의 SuperClass에 해당하는 XMI_header의 테이블생성이 이루어지고, 나머지 세 개의 SubClass중에서는 메타모델의 정보를 가지고 있는 XMI_content에 대한 테이블생성은 반드시 필요하지만 나머지 XMI_difference와 XMI_extensions의 데이터베이스 테이블은 경우에 따라서 생성하지 않는다.

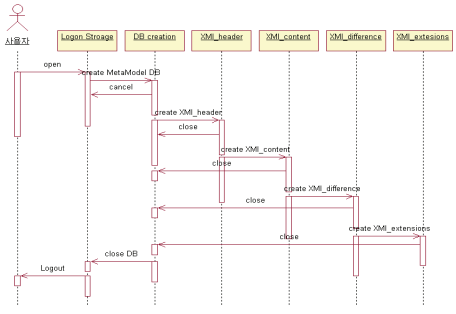

그림 2. XMI 메타모델 DB 생성 Sequence 다이어그램

3.2 XMI 메타모델 생성

일반적으로 객체지향모델링(object oriented modeling)에서는 UML을 이용한 방법이 많이 사용되고 있으

나[10], 본 논문에서는 UML에 의해 산출된 각 클래스 다이어그램의 형식들을 XML형식으로 자동 변환시키는 것을 표준화시키기 위한 방법으로 제안된 XMI(XML Metadata Interchange)를 이용하였다. 그래서 UML로 작성된 각종 다이어그램들은 XMI의 메타모델로 작성될 수 있으며, 또한 규정에 따라 XML로 표현될 수 있도록 하였다. 그리고 메타모델의 저장소 설계에서는 관계형 데이터베이스(relational data base)를 이용하여 메타데이터의 무결성을 위한 정규화가 적용된 테이블을 작성하였다.

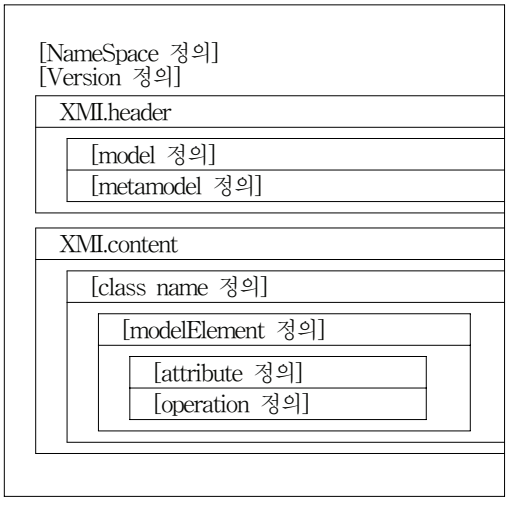

그림 3. 패턴메타모델 형식

3.3 XMI 관계형 데이터베이스 작성

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 4. XMI.header 테이블의 XMI 관계형 데이터베이스 모델

Ⅳ. XML 기반 클래스

4.1 XML 기반 클래스 설계

본 연구에서는 UML 모델링을 XMI 메타모델로 생성하고, 다시 메타데이터로 분리하여 단일 클래스 형식으로 DB에 저장하는 방법에 대해서 제시하였다. 이렇게 DB에 저장된 클래스의 메타데이터는 관계형 데이터베이스형의 메타데이터나 일반형의 XML 메타데이터로 변환이 가능하다. DB로 저장된 클래스 테이블에는 모델내의 클래스 상속관계를 표시하는 model name, SuperClass, SubClass등 필드가 있다. 여기서는 모델에서 클래스에 대한 관계뿐만 아니라 일반적인 단일 클래스의 메타데이터 생성에도 초점을 맞추어 XML 메타데이터 생성을 하였고, XML 메타데이터에 대한 마크업언어도 DTD기반이 아닌 XML 스키마 기반으로 하고있다.

DTD는 XML 문서구조를 표현하는 마크업언어를 작성하는 일반적인 방법으로 사용되어 왔으나, 세부적인 데이터구조를 표현할 수 있는 기능이 없다. W3C의 XML 스키마는 이와는 다르게 데이터의 구조에 대한 형식 및 제약사항등을 훨씬 효과적으로 표현할 수가 있어 XML 문서구조를 다양하게 정의하거나 검증하는데 사용이 일반화 되어가고 있다.

[그림 5]는 UML의 클래스 다이어그램을 이용하여 XML 스키마를 모델링 하고, 사용자들로 하여금 정확하고 효과적인 마크업언어를 작성할 수 있도록 하기 위해서 세 개의 단계로 나누어서 적용하고 있다. 첫 번째 단계는 UML 클래스 다이어그램을 이용한 메타모델 생성단계이며, 두 번째 단계는 UML 클래스에 대한 메타데이터를 생성하는 단계이고, 마지막 세 번째 단계는 마크업언어를 생성하는 단계이다.

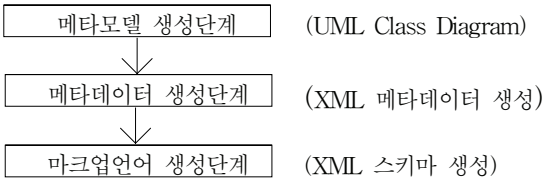

그림 5. 3단계 설계

4.2 XML 기반 클래스생성

본 연구에서는 UML에서 정의하고 있는 클래스에 대한 표기방법을 적용하기 위해 [그림 6] 을 가지고 메타데이터를 정의하였다. 여기서 Person은 클래스의 이름, personID은 어트리뷰트 그리고 PersonInfo()은 오퍼레이션이다. 일반적으로 클래스는 concrete 클래스와 abstract 클래스로 분류할 수 가 있다. concreate 클래스는 메소드가 구체화가 되어 있음을 나타내며, 객체지향에서는 인스턴스로 실체화가 가능한 클래스를 나타낸다. abstract 클래스는 추상화된 메소드를 나타내고 클래스의 이름을 기울임꼴로 표시한다. 클래스를 XML 메타데이터로 표시하는 엘리먼트에서는 <isAbstract> true </isAbstract>로 표시하여 abstract 클래스임의 나타내고, concrete 클래스인 경우에는 <isAbstract> false </isAbstract>로 표시한다.

그리고 UML에서는 속성(attribute) 및 오퍼레이션(operation)에 대하여 다음과 같은 세 개의 가시성(visibility)을 정의하고 있다.

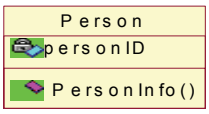

그림 6. 단위클래스

[그림 7] 의 단위클래스에 대한 데이터베이스 테이블 생성은 XMI 메타데이터 저장소 설계에 대한 제안방법을 적용하였다[14]. 그러나 여기서는 XMI 메타모델기반의 클래스에 대한 데이터베이스 생성하여 관계형 데이터베이스 형태로 메타데이터를 생성하였지만, 본 연구에서는 XML 메타데이터를 생성하고 있기 때문에 다음과 같이 대칭적으로 tag를 변형시켰다. 관계형 데이터베이스 메타데이터에서는 <Column> tag를 이용하여 엘리먼트의 어트리뷰트나 오퍼레이션의 속성에 대해서 일괄적으로 생성하고 있지만, XML 메타데이터에서는 이를 <attribute>와 <operation>으로 분리하여 각각의 속성에 대한 tag를 생성할 수 있도록 한 것이다.

<Table> -> <classDiagram>
<Table.name> -> <name>
<Column> -> <name>
<Column> -> <type>
<Column> -> <visibility>

	열 이름	데이터 형식	길이	Null 허용
🔑	class_name	char	20	
	isAbstract	char	10	
	att1_name	char	20	✓
	att1_type	char	20	✓
	att1_visibility	char	20	✓
	op1_name	char	20	✓
	op1_visibility	char	20	✓
	att2_name	char	20	✓
	att2_type	char	20	✓
	att2_visibility	char	20	✓
	op2_name	char	20	✓
	op2_visibility	char	20	✓
	att3_name	char	20	✓
	att3_type	char	20	✓
	att3_visibility	char	20	✓
	op3_name	char	20	✓
▶	op3_visibility	char	20	✓

그림 7. 단위클래스 테이블

[그림 8] 에서는 [그림 7] 의 단위클래스 테이블에 있는 각각의 칼럼기반으로 XML 메타데이터를 생성하는 형식을 정의한 것이다. 단위 클래스에 대한 데이터베이스 테이블 생성에서 클래스의 종류에 따라 어트리뷰트나 오퍼레이션의 개수에 차이를 둘 수 있지만, 본 연구에서는 응용보다는 정의에 대한 개념을 적용하기 위해서 각각 3개씩만 생성하였다.

```
<!-- version 지정 -->
<?xml version="1.0" encoding="버전"?>
<!-- Style sheet -->
<?xml-stylesheet type="text/xsl" href="스타일시트
"?>
<!-- Namespace 및 마크업언어 지정 -->
<uml xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance
" xsi:noNamespaceSchemaLocation = "마크업언어"
>
  <classDiagram>
    <class classID="클래스 이름">
      <name>클래스 이름</name>
      <isAbstract>추상화클래스확인</isAbstract>
        <attribute>
        <name>어트리뷰트 이름</name>
        <type>어트리뷰트 데이터타입 </type>
        <visibility>어트리뷰트 가시성</visibility>
      </attribute>
      <operation>
        <name>오퍼레이션 이름</name>
        <visibility>오퍼레이션 가시성c</visibility>
      </operation>
    </class>
  </classDiagram>
</uml>
```

그림 8. XML 메타데이터 형식

4.3 XML 스키마언어 데이터타입 분류

앞에서 정의한 클래스의 XML 메터데이터의 마크업 언어 정의에 사용되는 컴포넌트의 형식 중 본 논문에서는 심플타입(simple type)과 콤플렉스 타입(complex type)으로 분류하였다. XML 스키마에서 데이터타입은 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 나눌 수 있다. 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>
```

그림 9. 글로벌 심플타입 정의

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

그림 10. 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 <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 11. 순차적 자식 엘리먼트 콤플렉스 타입

<sequence>의 자식 엘리먼트로 대개 여러개의 엘리먼트가 오지만 한 개의 엘리먼트만 사용되는 경우에도 반드시 <sequence> 엘리먼트를 삽입해야 한다. 그리고 "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 12. 선택적 자식 엘리먼트 콤플렉스 타입

V. XML 스키마언어 생성도구 설계 및 구현

[그림 13] 은 사용자 정의 심플타입 XML 스키마언어를 생성하기 위한 중요한 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 13. 심플타입 XML 스키마언어 생성 알고리즘

[그림 14] 는 심플타입 XML 스키마언어 생성방법에 대한 프로그램정의를 이용하여 설계한 GUI 인터페이스이다. 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.

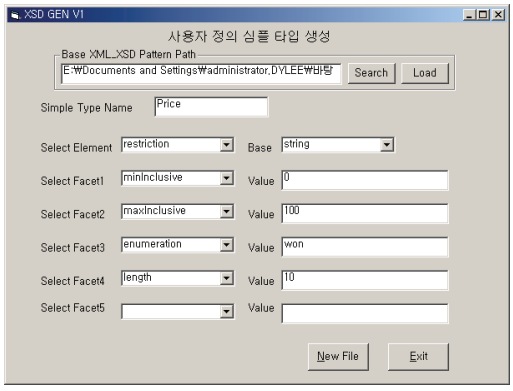

그림 14. 사용자 정의 심플타입 인터페이스

그리고 사용자 정의 콤플렉스타입 XML 스키마언어를 생성하기 위한 방법으로 [그림 15] 의 알고리즘에서는 요구되는 각 엘리먼트들을 단계적으로 생성할 수 있도록 하였다. 기본적인 생성방법은 [그림 13] 과 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 15. 사용자정의 콤플렉스타입 알고리즘

[그림 16] 의 사용자정의 콤플렉스타입(선택적자식엘

리먼트) XML 스키마언어 인터페이스는 [그림 15] 의 프로그램의 작성알고리즘에 의해 생성된 것이다. 여기서는 엘리먼트의 속성으로 DTD에서 지원하지 못하는 광범위한 데이터 타입을 정의하게 하고, 최소발생횟수와 최대발생횟수에 대한 속성을 추가하였다.

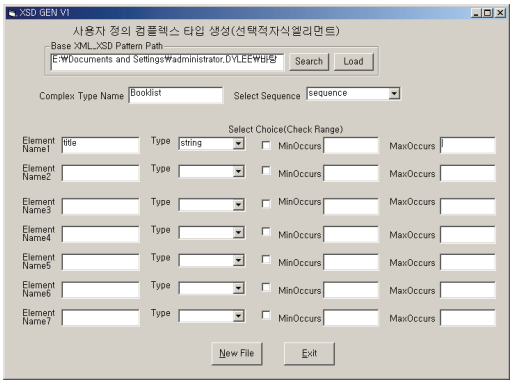

그림 16. 사용자정의 콤플렉스타입 인터페이스

VI. 평가 및 결론

XMI 메타모델과 XML 메타데이터를 이용하여 클래스사용에 대한 연구는 일반적으로 사용되고 있는 XML 메타데이터의 생성도구와 상당한 차이점을 가지고 있다. 현재 사용되는 XML 문서도구 시스템은 에디터기능과 데이터베이스 연동, 다른 객체지향 언어로의 변환,

표 1. 제한적 메타모델적용기반 시스템비교

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

그리고 마크업언어의 생성부분에 많은 비중을 두고 개발하고 있다. 그러나 본 연구는 기존 시스템에서 지향하는 부분보다는 XMI 메타모델에서 추출되는 클래스의 메타데이터의 마크업언어를 조건에 따라 생성하는데 초점을 맞추었다. 그래서 툴(tools)을 이용한 일반적인 메타데이터의 마크업생성에 대한 접근보다는 매우 제한적으로 본 연구에서 제안하고 있는 부분에 국한하여 비교되었다.

[표 1] 의 비교에서 알 수 있듯이 오스트리아 빈 공대에서 개발한 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. ALTOVA에서 개발한 XMLSPY는 현재 XMLSPY2004 Ent버전까지 출품되었으며, 이것은 XML 전문에디터로 웹 서비스 기술, WSDL, XSL/XSLT, SOAP, XML 스키마 등 XML과 관련이 있는 어플리케이션을 설계하고 디버깅하기 위한 표준 XML 기반의 환경을 가지고 있고, 생산성과 가치성이 매우 우수하다. 그러나 XML SPY 역시 본 연구에서 중점을 두고 있는 다양한 데이터타입의 지원에 대해서 심플타입과 콤플렉스타입을 이용한 스키마 생성이 가능하지만 본 논문에서 연구한 XMI 메타모델기반의 메타데이터 마크업언어의 생성에는 적절하지 못한 일반적인 XML 메타데이터 기반의 마크업언어 생성에 기반을 두고 있다. 그리고 XML SPY 등 GUI 환경의 트리모형으로 그래픽 화 되어있는 툴을 이용하여 마크업언어를 생성하는 것은 사용자들에게 매우 쉽고 간단하게 이용할 수 있는 장점이 있지만 클래스내의 단위 엘리먼트의 속성을 부여할 수 있는 방법이 없기 때문에 모델내의 클래스의 관계를 표현할 수 있는 SuperClass와 SubClass에 대한 적절한 타입의 속성을 표현할 수 없다. 그러나 엘리먼트에 대한 어트리뷰트를 표현하는데 다양하고 세부적인 데이터타입이 지원되므로 일반적인 메타데이터기반의 마크업언어 생성에서는 매우

유용하게 사용할 수 있다. 그리고 메타데이터에 대한 전체의 마크업언어를 생성할 수 있어 영역별 재사용을 위한 분리된 언어의 생성이 쉽지 않다.

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 마크업언어 생성을 연구하였다. 그리고 트리형태의 마크업언어 생성보다는 쉽게 접근할 수는 없지만 XMI 메타모델기반의 클래스에 대한 메타데이터 마크업언어를 다양화되어 있는 엘리먼트타입와 어트리뷰트의 속성을 세부적으로 작성할 수 있도록 하였다.

마지막으로 본 연구에서는 메타모델과 메타데이터에 대한 부분적인 연구만 이루어지고 있어, 장기적으로 Rational Rose와 같은 도구(tools)에 의해 도식화된 패턴이나 클래스 다이어그램에 대한 메타모델 및 메타데이터, 관계형 데이터베이스 작성, 마크업언어 생성 등 이와 관련된 것들에 대한 연계적인 자동화 도구 개발이 요구되어 진다.

참 고 문 헌

[1] Gregor Engels and 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] Michael Thomsen and Michael. "*Creative Object Oriented Modelling Department of Computer Science,*" University of Aarhus, Aabogade 34, 8200 Aarhus N, Denmark..www.ideogramic.com/download/resources/ecoop2000.pdf.

[3] "OMG Unified Modeling Language Specification (draft)" Version 1.3. beta R7, 1999(6).

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

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

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

[7] 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.

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

[9] M.wein, S.MacKay, W. Gentleman, "Evolution is Essential for Software Tool Development," IWCASE '95, pp.196-205

[10] 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, pp.317-321, 2002.

[11] "XMI Gets the Capability to convey information," 1999.

[12] Georg Gottlob, Micheal Schrefl, and Brigitti Rock, "Extending Object-Oriented Systems with Roles", ACM Transactions on Information Systems 14, 3, pp.268-296, 1996.

[13] 최한용 "*XMI기반의 디자인패턴 설계 및 지원환경 구축*" 경희대학교 대학원 전자계산공학과, 박

사학위논문, 2002.

[14] 이돈양, “XMI 기반 객체지향 메타모델 생성”, 정보처리학회논문지D 제11-D권 제2호, pp.397-406, 2004.

[15] R. Conrad, D. Scheflner, J.c. Freytag : XML Conceptual Modeling Using UML, Proc, Conceptual Modeling Conference ER 2000, Salt Lake City, USA, Springer Verlag, pp.558-571, 2000.

[16] C. Kleiner and U. Liepeck : Automatic generation of XML-DTDs from Conceptual database schemas, Datenbank-Spektrum 2, dpunkt-Verlag, pp.14-22, 2002.

[17] Moh C, H., Lim E: p., and Ng W. K, Re-engineering Structures from web Documents. In ACM Digital Libraies 2000, San Antonio, Texas, USA, pp.67-76, 2000(6).

[18] Minos N. Garafalakis, Aristides Gionis, Rajeev Rastogi, S. Seshadri, and Kyuseok Shim. XTRACT : A System for Extracting Document Type Descriptions from XML Documents. In Proc. ACM SIGMOD, Dallas, Texas, USA, pp.165-176. ACM, 2000.

저 자 소 개

이 상 식(Sang-Sik Lee) 정회원

- 2000년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2004년 8월 : 경희대학교 전자계산공학과(공학박사)
- 2001년 3월 ~ 현재 : 송호대학 보건의료기기과 교수

<관심분야> : 소프트웨어 신뢰성, XML

최 한 용(Han-Yong Choi) 정회원

- 1998년 2월 : 경희대학교 전자계산공학과(공학석사)
- 2002년 8월 : 경희대학교 전자계산공학과(공학박사)
- 2004년 3월 ~ 현재 : 한북대학교 컴퓨터정보학과 교수

<관심분야> : 컴포넌트디자인, XML