변환정보파일을 이용한 XML Schema 자동생성 시스템 설계 및 구현

김서강*, 이언배
한국방송통신대학교 정보과학과
e-mail : sagaoi@lycos.co.kr*, lub@mail.knou.ac.kr

Design and Implementation of XML Schema Generation System based on Transfer Information File

Seo-Gang Kim*, Eun-Bae Lee
Dept. of Computer Science, Korea National Open University

요 약

XML 문서의 구조를 정의하는 XML Schema 는 문법이 복잡하며 설계자에 따라 표현방법이 달라질 수 있고 잘못된 설계는 시스템의 자원을 낭비할 수 있다. 특히 한정된 시간과 비용으로 진행되는 프로젝트에서 잘못된 XML Schema 설계는 프로젝트의 지연 및 비용 초과의 주된 원인이 된다. 따라서 본 논문에서는 XML 로 작성된 데이터 중심의 전자문서 전송 시 필요한 XML Schema 를 표준화된 방식으로 생성하는 변환정보파일 모델을 설계하였다. 그리고 설계된 변환정보파일모델에 정의된 메타 데이터를 이용하여 객체지향 개념을 도입한 XML Schema 를 자동 생성하는 시스템을 설계, 구현하여 표준화된 XML Schema 를 생성하였다.

1. 서론

다른 시스템간의 데이터 교환은 모두 사전에 정의되어진 형식의 전자문서를 통해 이루어진다. 하지만 거대한 전산시스템의 일부분으로 특정 업무를 처리하는 시스템들은 각각 주어진 업무를 처리하기 위해 필요로 하는 전자문서형식이 틀릴 수 있고, 특히 그것이 타 시스템인 경우는 말할 나위가 없다. 인터넷의 발달로 XML 이 여러 산업분야에서 전자문서의 대표적인 방법 중 하나로 사용되며 급속히 확산되고 있다. 최근 EAI(Enterprise Application Integrety), B2B, 웹서비스, SOA (Service Oriented Architecture) 등 산업계의 다양한 분야에서, 자체적으로 사용하고 있던 XML 문서를 표준화하여 비즈니스 파트너들간에 공용하기 위해 DTD 나 XML Schema 를 통해 XML 문서를 정의하고 있다. 이렇게 정의한 것을 "협의적인 레벨(the Agreements level)"의 Metadata XML Standard 라고 하며, 기업간의 상호 운영성을 증가시키는 데는 필수적이다.[1]

XML 문서의 구조를 정의하는 DTD 는 다양한 데이터를 표현하고 정확한 문서 구조를 정의하기에는 많은 문제점이 있어 산업 여러 분야를 만족시키기 어렵다. W3C 의 권고안[2,3,4]으로 채택된 XML Schema 는 DTD 보다 표현력이 풍부하고 정확한 자료 구조를 제공하며 자기 기술적이고 XML 로 표현되어있다.

본 논문에서는 XML 로 작성된 데이터 중심의 전자문서 전송시 변환정보파일에 정의된 메타 데이터를 이용하여 객체지향 개념을 도입해서 해당문서의 구조를 표현할 때 필요한 XML Schema 를 표준화된 방식으로 자유롭게 생성한다. 또한 생성된 XML Schema 를 이용하여 XML 전자문서 작성시의 유효성 검사 및 네트워크를 통해 수신된 XML 전자문서를 응용프로그램이 처리하기 전의 구조적인 유효성 검사를 효율적으로 수행하여 응용프로그램에 의한 사전 Validation 검증을 최소화한다.

본 논문의 구성은 다음과 같다. 2 장에서는 XML Schema 자동생성과 관련된 연구들을 살펴본다. 3 장에서는 변환정보파일을 이용한 XML Schema 자동생성 시스템의 전체적 구조를 보여준다. 또한 변환정보파일의 구조 및 정의방법, XML Schema 자동생성기의 설계과정을 설명하며 구현한다. 마지막으로 4 장에서는 본 논문의 결론 및 향후 연구과제를 기술한다.

2. 관련연구

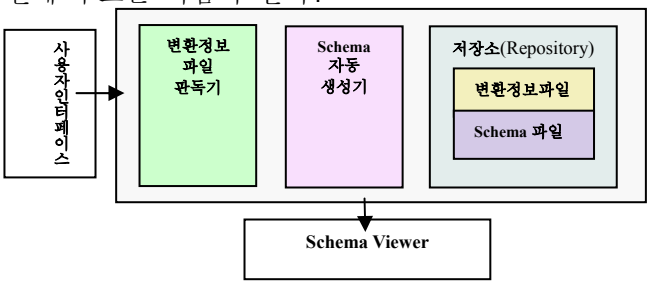
XML Schema 를 자동 생성하는 시스템은 최근까지 보편화 되어 있지 않고 전문적인 XML Schema 설계자에 의한 설계에 의존하고 있다.[5][6] 현재 나와있는 XML Editer 관련 툴이나 연구들은 XML Schema 를 자동 생성할 수는 있지만, 이미 정의되어진 DTD 파일을 이용하여 XML Schema 로 자동 변환하거나 사전에 작성된 XML 파일에 정의된 값들의 정보를 추출하여 XML Schema 를 생성[7][8]하게 된다. 이렇게 생성된 XML Schema 는 다양한 데이터를 표현하고 정확한 문서구조를 정의[9]하는 XML Schema 의 특징을 제대로 이용하지 못한다. 또한 포괄적인 범위의 XML Schema 파일을 생성하게 되며, 해당 XML 전자문서에 대한 상세한 Validation 검증을 내부 어플리케이션 프로그램을 통해 또 한번 거쳐야 하는 단점이 있다. 기존의 자동생성 시스템들은 사용자가 각 요소별로 요소명, 데이터 타입, 최대값, 최소값 등의 요소별 속성들을 사용자 인터페이스를 통해 직접 입력하거나 선택하여 각각의 요소에 대한 스키마 객체를 생성해낸 후 이를 전체적으로 조합하여 생성되는 구조[10]로 구성되어 있다.

3. 변환정보파일을 이용한 XML Schema 자동생성스템

데이터 중심의 데이터 전송과 관련된 XML 전자문서의 경우 문서의 구조나 정의된 항목의 타입은 복잡하지 않지만, 항목이 너무 많거나 사전에 정의된 항목이 추가 또는 변경되는 경우가 자주 발생한다. 항목별로 스키마를 생성하는 기존의 자동생성시스템은 항목이 추가되거나 순서가 변경될 경우 단지 추가 또는 변경된 항목뿐만 아니라 처음부터 다시 순차적으로 입력해야 하는 비효율성이 있다. 또한 표준화된 설계 규칙 없이 설계자에 의해 독자적으로 설계, 개발되는 XML Schema 는 시스템의 성능을 저하시킬 수 있으며 상호연동시에 많은 장애를 초래할 수 있다.[1] 따라서 본 장에서는 B2B, Web Service, SOA(Service Oriented Architecture)등에서 데이터 중심의 XML 전자문서를 상호 교환할 경우 사전에 정의되는 전자문서 구조에 XML Schema 생성을 위한 몇몇 항목을 추가한 변환정보파일을 제안한다. 또한 제안된 변환정보파일을 이용한 XML Schema 자동생성기를 객체지향방법론을 도입하여 설계, 구현한다.

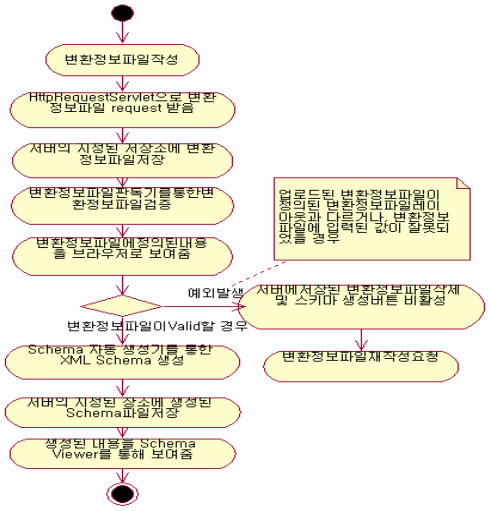
3.1 XML Schema 자동생성시스템의 전체 구조

본 논문에서 설계한 XML Schema 자동생성시스템의 전체 구조는 다음과 같다.



[그림 1] XML Schema 자동생성시스템 전체구조

변환정보파일을 작성하여 변환정보파일을 등록하면 변환정보파일 판독기가 해당 파일을 읽어들인다. 변환정보파일 판독기는 사전에 정의되어진 변환정보파일양식과 읽어 들인 변환정보파일을 비교한다. 읽어들인 변환정보파일의 오류가 없는지 검토한 후, 변환정보파일에 정의되어 있는 해당 내역을 그대로 브라우저에 보여준다. 만일 변환정보파일에 작성된 내역이 오류가 있을 경우 Schema 자동생성버튼이 활성화되지 않고 오류가 있는 행을 반전된 색으로 나타내어 변환정보파일을 다시 작성하여 등록하도록 한다. 변환정보파일 판독기를 제대로 통과할 경우 Schema 자동생성버튼이 활성화되며, Schema 자동 생성기를 통해 해당 변환정보파일의 양식에 맞게 XML Schema 가 자동 생성된다. 생성된 XML Schema 는 Schema Tree Viewer 화면을 통해 보여주며 지정된 Repository 에 변환정보파일과 함께 저장된다.



[그림 2] XML Schema 자동생성시스템 Activity Diagram

3.2 변환정보파일의 구성

일반적인 XML 전자문서 변환 어플리케이션은 일정한 저장소에 저장된 해당전자문서의 구조에 대해 작성된 DTD 또는 Schema 파일을 읽어 출력전자문서에 대한 형식을 분석한 다음, 자동으로 출력전자문서를 생성해낸다. 하지만 전자문서 형식이 다양한 여러 업무의 추가 및 변경이 빈번한 시스템에서는 저장된 DTD 나 XML Schema 는 끊임없이 전문형식이 변경될 때 마다 수정을 가해야만 한다. 하지만 변환정보파일을 이용하여 Schema 를 자동생성 할 경우, 전문변환 어플리케이션의 소스파일 수정 및 컴파일 과정 없이 단지 변환정보파일의 추가 및 변경으로 새로운 업무처리가 가능하게 되어, 운영관리 및 위험요소 측면에서 보다 효율적이다. 기업간 데이터 전송과 관련된 전문구조를 분석하여 보면, 데이터 중심일지라도 단순한 나열식 구조가 아닌 2 차원 배열 이상의 형태에 대한 전문구조를 접할 수 있게 된다. 따라서 본 연구에서는 2 차원 배열 이상의 구조를 갖는 복잡한 전문형태에 대해서도 자동으로 XML Schema 를 생성할 수 있도록 변환정보파일을 설계한다. 또한 변환정보파일로

는 전자문서를 정의할 때 기업간에 보편적으로 통용되는 엑셀시트를 이용한다. 일반적인 전자문서 양식에서 XML Schema 를 정의하기 위해 필요한 몇몇 항목을 추가하여 변환정보파일을 다음과 같이 정의한다.

No	분류 1	2	3	영문명	정의	유형	길이	발생빈도	참조	패턴
1	Y			Header	헤더정보			1..1	N/A	
2	Y	Y		DocumentId	해당문서 번호	S	8	1..1	N/A	
3	Y	Y		DateTime	해당문서 전송일	T	35	1..1	N/A	
4	Y	Y		SupplierParty	구매업체정보			1..2	N/A	
5	Y	Y	Y	Party.Identifier	구매업체 식별자	S	14	1..1	N/A	₩d{6}-₩d{7}
..	...	..	...	.....					N/A	
9	Y			Line	라인정보			1..1	N/A	
10	Y	Y		OrderItem	주문내역			1..3	N/A	
11	Y	Y	Y	ItemId	거래항목 코드이름	S	17	1..1	N/A	
12	Y	Y	Y	Item.Name	거래항목 이름	S	35	1..1	N/A	
13	Y	Y	Y	Item.Quantity	거래항목 수량	N	18	1..1	N/A	
..	...	..	...	.....					N/A	
23	Y			Response.Code	응답 유형 코드	S	4	1..1	N/A	

[그림 2] 변환정보파일의 구성

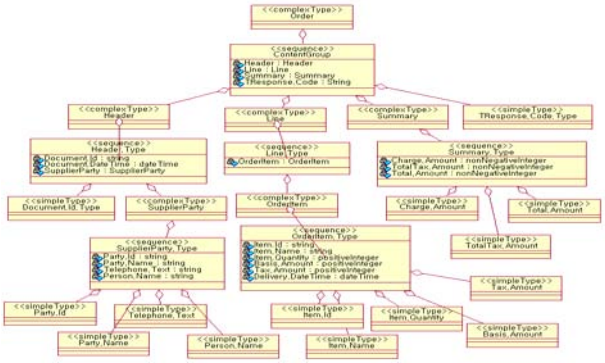
위에서 제시한 변환정보파일에 대한 각 항목별 의미는 다음과 같다.

매핑파일	설명
순번	순번에 따라 순차적 트리 구조로 XML Schema 를 생성한다.
분류	해당 엘리먼트가 2 차원 또는 3 차원 배열구조로 정의되는지를 판별한다.표현값은 'Y'이며, 분류 1 에 'Y'값이 없으면서 분류 2 에 'Y'값을 정의할 수 없다. 분류 3 에 'Y'값이 정의되기 위해서는 반드시 분류 1,2 에 'Y'값이 정의되어 있어야 한다.
영문명	정의된 영문 항목명은 XML Schema element 의 속성 name 의 값이다.
정의	생성되는 XML Schema element 의 주석으로 사용된다.
유형	정의하는 XML Schema element 데이터 타입을 지정한다. XML Schema 에 정의되어 있는 기본 데이터 타입 및 사용자 정의 타입, 복잡 타입을 모두 지원하도록 한다. 만일 정의된 값이 없으면서 분류 1 또는 분류 1,2 에 'Y'로 정의되어 있으면 해당항목의 요소 타입은 ComplexType 으로 지정되며 하위 행들을 자식으로 갖게 된다.
길이	입력 데이터 값의 길이 제한을 정의한다. 만일 유형에 기본 데이터 타입 값이 정의되어 있으면서 길이 항목에 값이 입력되지 않을 경우 해당되는 요소의 데이터 타입은 기본 데이터 타입이 된다.
발생빈도	정의한 항목 요소의 최소 및 최대 발생횟수이다. 속성 maxOccurs 와 minOccurs 의 값으로 매핑된다.
참조	외부에 정의된 Schema 파일을 참조할 때 사용된다.
패턴	주민등록번호와 같은 고정된 패턴을 값으로 가지는 사용자 정의 simpleType 을 정의할 때 표현되는 정규식을 정의한다.

순번에 따라 순차적으로 XML Schema 를 구성하되, 유형의 값이 지정되지 있으면서 분류의 1, 또는 2 항목 값이 'Y'로 정의되어 있으면 해당 항목의 element 타입은 ComplexType 으로 설계하며 각 항목에 대한 요소의 type 은 기본값을 SimpleType 으로 정의한다. 또한 발생빈도에 정의된 값 만큼 최대 반복이 가능하며 값이 기재되지 않을 경우 기본값을 1 로 설정한다.

3.3 XML Schema 자동 생성기 설계

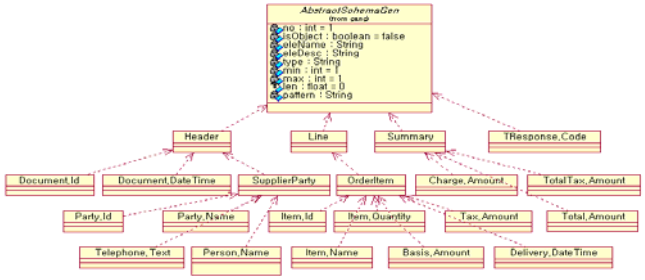
최상위 요소를 ROOT 으로 지정하여 정의된 각 행들의 값이 변환정보파일의 각 항목별 속성들을 정의한 AbstractSchemaGen Class 를 상속하는 각각의 객체로 생성되어 해당정보를 저장하며 이차원배열에 순차적으로 담겨진다. 이 때 반복구간이 있을 경우 해당 반복구간 전체를 하나의 객체로 생성하여 한 배열에 정의한다. 처음 생성된 Schema 는 단순히 ROOT 요소의 하위요소로 ref 속성에 의해 참조 가능하게 정의되어진다.



[그림 3] Schema 와 UML 표기

반복구간이 있는 행들은 위에서 정의된 객체를 상속받아 하위객체를 생성하며 최대반복횟수를 maxOccurs 속성을 이용하여 나타낸다. 배열에 담긴 각 객체들을 꺼내어서 하위요소를 가지지 않을 경우 정의된 변환정보파일에 맞게 해당 요소를 작성하여 준다.

```
<xsd:element name="Telephone.Text" type="Telephone.Text.type" />
<xsd:element name="Person.Name" type="Person.Name.Type"/>
<xsd:simpleType name="Telephone.Text.type">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="₩d{2,3}-₩d{3,4}-₩d{4}"/>
    </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="Person.Name.Type">
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="15"/>
    </xsd:restriction>
 </xsd:simpleType>
```



[그림 4]변환정보파일에 정의된 각 행별 객체생성 Class Diagram

만일 하위 객체를 가질 경우 순차적으로 최하위객체에 정의된 값들을 먼저 작성한 다음 다시 상위로 올라와서 해당 요소들을 작성하여준다.

```
 <xsd:complexType name="Header.Details">
  <xsd:sequence>
    <xsd:element name="Document.Id" type="Document.Id.Type"/>
    <xsd:element name="Document.DateTime" type="xsd:dateTime"/>
    <xsd:element name="SupplierParty"  type="SupplierPartyType.Details"
maxOccurs="2"/>
  </xsd:sequence>
 </xsd:complexType>
<xsd:complexType name="SupplierPartyType.Details">
  <xsd:sequence>
    <xsd:element name="Party.Identifier" type="Party.Identifier.Type"/>
    <xsd:element name="Party.Name" type="Party.Name.Type"/>
    <xsd:element name="Telephone.Text" type="Telephone.Text.type"/>
    <xsd:element name="Person.Name" type="Person.Name.Type"/>
  </xsd:sequence>
 </xsd:complexType>
```

[Schema 자동 생성 알고리즘]

```
입력:변환정보파일
출력:XML Schema
ArrayList arry = new ArrayList();
변환정보파일에 정의된 행만큼 반복하면서 객체생성하기.
for(int i=0; i<순번; i++){
```

```
 if(배열.equals("")|| 유형.equals("None") )
    array.add(createClass());
  else { //1차 반복구간이 있을 경우
    ArrayList array1 = new ArrayList();
    if(배열.equals("Y")|| !유형.equals("None") )
        array1.add(createClass());
  }
}
//각각의 객체를 꺼내 정의된 내용에 맞게 Schema 생성하기
for(int i=0; i<순번; i++){
   if("None".equals(array.get(i).type){
     makeComplexType();
    }
else{  makeSimpleType();}
```

3.4 구현 및 평가

구현에서는 사전 전제조건으로 XML 전자문서의 속성을 배제하며 일반적으로 사용되는 문자, 정수, 실수형에 한하여 2 차원 이상의 배열구조의 반복구간이 있는 그림 2 에서 제시된 변환정보파일을 예로 사용하였다. 그림 5 는 변환정보파일을 읽어 XML Schema 를 자동 생성하는 사용자 인터페이스로 그림 2 의 변환정보파일을 읽어 사전 Validation 을 수행한 모습이다.

[그림 5] 변환정보파일을 읽어들이는 사용자인터페이스

그림 6 은 그림 5 의 사용자 인터페이스 화면에서 보여준 변환정보파일에 정의된 메타 데이터를 확인한 후 스키마 자동생성기를 통하여 생성된 XML Schema 이다.

[그림 6] Schema 자동생성시스템에 의해 생성된 Schema

생성된 XML Schema 는 자동생성시스템에 의해 표준화된 방식으로 생성되어 설계자에 따른 서로 다른 표현방법이 사전에 배제되었으며, 잘못된 설계를 미리 방지할 수 있었다. 또한 변환정보파일을 이용함으로써 누구나 손쉽게 XML Schema 생성이 가능하여 프로젝트에서 시간과 비용의 효과를 가져올 수 있었다.

4. 결론 및 향후 과제

XML 문서의 구조를 정의하는 XML Schema 는 문법이 복잡하며 설계자에 따라 표현방법이 달라질 수 있어 일괄성을 지니지 못하는 문제점이 있었다. 또한 잘못된 설계는 시스템의 자원을 낭비할 수 있으며, 특히 한정된 시간과 비용으로 진행되는 프로젝트에서 프로젝트의 지연 및 비용 초과의 주된 원인이 되었다.

본 논문에서는 사전에 정의된 변환정보파일을 이용하여 XML Schema 를 생성함으로써 일관된 설계 규칙에 의해 표준화된 XML Schema 를 생성하였다. 제시한 변환정보파일은, 새로운 업무가 빈번하게 추가되거나 정의된 항목의 속성, 발생빈도, 길이, 문서 내 항목의 위치가 변경될 경우 변환정보파일만 추가 또는 수정하면 되기 때문에 시스템의 변경이 전혀 필요 없고, 시스템의 변경에 따른 위험 요소가 감소한다. 또한 XML Schema 자동 생성기는 숙련된 XML 기술이 필요하여 반복적이고 오류가 발생하기 쉬운 일련의 과정들을 사라지게 하며, XML Schema 의 복잡한 문법구조를 알지 못하더라도 정형화된 양식의 XML Schema 설계가 가능해졌다. 생성된 XML Schema 는 일관된 설계구조로 인해 웹 환경과의 융합 및 웹 환경으로의 확장이 용이하다.

향후 연구과제로는 현재 데이터전송중심의 XML 문서뿐만 아니라 좀더 다양하고 복잡한 구조로 작성되는 문서중심의 XML 문서에 대해 보다 효율적으로 XML Schema 를 자동 생성할 수 있는 시스템의 설계로 연구를 확장하여 진행할 예정이다.

참고문헌

[1] Realizing XML benefits in Life Insurance, Umesh Kumar Rai, Business Analyst, Insurance Practice

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

[3] W3C, "XML Schema Part 1: Structures", http://www.w3.org/TR/xmlschema-1, 2 May.2001.

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

[5] 채진석(외 6),한국전자거래진흥원, XML 스키마 설계 및 XML 문서 설계 연구 및 최적안 도출, 2002.12

[6] 윤영미,황석형,양해술,생체인식 소프트웨어의 평가모듈을 위한 XML 스키마 설계, 제 22 회 한국정보처리학회 추계학술발표대회 논문집 제 11 권 제 2 호, 2004.11

[7] 정성윤, 김성진, 비정형 기술문서에 대한 XML 스키마 개발, 제 22 회 한국정보처리학회 추계학술발표대회 논문집 제 11 권 제 2 호, 2004.11

[8] 조정길, 구연설, 스키마가 없는 XML 문서에서의 재사용 가능한 XML Schema 추출 기법, 정보처리학회논문지 D 제 10-D 권 제 4 호, 2003.08

[9] 신민철, XML 웹서비스, FREELEC, 2004.04

[10] 김동영, XML/EDI 시스템 구축을 위한 XML Schema 자동생성 시스템 구현, 정보기술연구소 논문지, 동아대학교, 2000.