Xvert :재사용 가능한 XML Schema 자동 변환기 설계

고혜경[0] 조윤기 조정길 이병렬 구연설
충북대학교 전자계산학과 소프트웨어공학 연구실
{ellefgt, ykcho}@selab.chungbuk.ac.kr}

cho0530@chollian.net

inanet@splinux.co.kr

yskoo@cbucc.chungbuk.ac.kr

Design of Reusable Automatic Translator for converting DTD to Schema

Hye-Kyung Ko[0] Yun-Kee Cho Jung-Gil Cho Byung-Real Lee Yeon-Seal Koo
Dept. of Computer Science, Chungbuk National University

요 약

최근 인터넷상에서 DTD 기반의 XML 문서가 문서의 교환의 목적으로 사용되어 B2B 상에서 XML 문서의 사용이 증가하고 있다. 그러나 DTD는 데이터 타입이 제한적이고 사용자가 원하는 형태를 정의하여 사용할 수 없기 때문에 좀 더 유연하고, 재사용이 가능한 스키마를 B2B 상에서 표준으로 이용을 하게 되었다. 스키마는 객체 지향적이기 때문에 새로운 타입을 선언 시 기존의 데이터와 타입을 이용하여 확장 또는 제한을 하여 재사용성이 뛰어나다. 그러나 기존의 스키마 자동 생성기는 사용자가 바라는 최적화 코드가 아니고, 스키마의 장점인 모듈화가 전혀 되어 있지 않아, 재사용을 할 수가 없다. 따라서 본 논문에서는 기존 자동 생성기의 단점을 극복하여 문서를 XML 스키마로 변환할 때 모듈화가 되고, 재사용이 가능한 코드를 만들 수 있도록 자동 변환기를 설계하며, 기존의 변환기가 사용자 시점에서 사용하기가 어렵고, 복잡한 인터페이스를 가지고 있기 때문에 이를 사용하기 쉽게 단순화하고 DTD와 스키마를 다양한 외양으로 볼 수 있는 스타일(Style) 에디터를 첨가하여 변환기를 구성한다.

1. 서 론

인터넷 비즈니스에서 XML 문서는 정보 통신망의 발달과 함께 문서 교환의 표준으로 자리잡고 있다. 최근 인터넷상에서 DTD 기반의 XML 문서가 문서의 교환의 목적으로 사용되어 B2B 상에서 XML 문서의 사용이 증가하고 있다. 그러나 B2B 상에서 문서교환시스템은 각기 다른 형태의 문서구조를 가지고 있어서 시스템의 통합 및 상호운용에 많은 어려움이 있다. 그 중에서 XML 문서를 다른 XML 문서로 변환하는 변환기가 필요하다[1][2]. 그러나 DTD는 다양한 데이터 타입을 정의할 수 없기 때문에 데이터베이스에 매끄럽게 연계시키기가 어렵고, 사용자가 원하는 형태를 정의하여 사용할 수 없기 때문에 유연성이 떨어지며, 재사용을 할 수가 없다.[2][4] 이에 본 논문에서는 XML DTD 문서를 스키마 문서로 변화하는 XML 스키마 자동 변환기를 설계한다. 본 논문에서 제안하는 XML 스키마 자동 변환기는 스키마의 장점인 모듈화와 사용자의 요구 사항을 잘 분석해 재사용이 가능한 코드를 생성하여 유연성이 높은 구조를 가진 시스템이 되도록 자동 변환기를 설계하고, 기존의 변환기가 가지고 있는 DTD 에디터와 XML 문서를 다양한 외양으로 볼 수 있는 스타일 에디터를 첨가하여 구성을 한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 본 논문과 관련되어 기존에 작성된 스키마 자동변환기를 살펴보고, 3장에서는 XML의 개요와 DTD를 스키마로 변환하는 변환 규칙을 살펴보며 4장에서는 본 논문에서 제안하고 있는 자동변환기의 전체적인 시스템의 흐름과 각 요소의 설계를 보여준다. 5장에서는 결론 및 향후 연구를 제시한다.

2. 관련 연구

본 논문에서는 현재 나와 있는 DTD를 스키마로 변환하는 자동 변환기들 중에서 Icon Information Systems사의 XML Spy 3.0과 Turbo XML v2.2 두 가지 변환기를 소개를 한다.

2.1 XML Spy 3.0

Icon Information Systems에서 개발한 XML Spy 3.0의 특징은 편집화면을 Grid View 형태로 보여주는 것이다. 일반적인 XML 에디터들은 Tree 형식의 View를 제공하는데, XML Spy의 경우에는 Tree 형식의 view를 Grid에 구현하였다. XML Spy는 많은 형식의 파일들을 편집할 수 있으며, Schema 편집이 가능하다. 또한 project관리 개념을 가지고 있어서 대량의 파일들을 편집할 때에 사용하면 편리하다.

그러나 DTD와 스키마 사이의 변환이 단지 클릭 몇 번으로 원하는 것으로 변환을 할 수가 있지만 자동으로 생성된 코드가 갖는 공통적인 문제로, 사용자가 바라는 최적화 코드를 생성할 수가 없고, 처음 사용하는 사용자는 사용법을 쉽게 알 수가 없고 사용 방법이 복잡하다.[7]

2.2 Turbo XML v2.2

Turbo XML v2.2는 XML DTD, 스키마, XML 다큐먼트를 변환해주는 툴로써, 다이어그램을 그래픽하게 보여주기 때문에 DTD와 스키마의 변환 과정을 시각적으로 볼 수가 있다. 그러나 다른 에디터들과 마찬가지로 초보자가 사용하기가 어렵고, 불필요한 내용이 프로그램에 너무 많아 사용하는데 복잡하고, 다이어그램을 통해서 변환된 모습을 보여주기 때문에 엘리먼트 간의 구조를 보기가 나쁘다.[6]

2.3 Xvert : XML 스키마 자동 변환기

XML Spy와 Turbo XML v2.2 가 DTD를 스키마로 변환할 때, 변환된 스키마가 사용자가 원하는 코드가 아니고, 최적의 코드를 만들기 위해서 다시 한번 사용자의 수 작업이 필요하기 때문에 사용자의 수 작업을 더는 최적화 코드를 생성하고, 스키마의 특징인 재사용을 할 수 있도록 코드를 생성하도록 본 논문에서는 스키마 자동변환기를 설계한다. 또한 처음 사용하는 사용자도 변환기를 사용할 수 있도록 사용자 인터페이스에서 사용하기 쉽고, 최소한의 변환기능만을 제공한다.

3. DTD를 스키마로 변환하는 변환 규칙

DTD와 스키마는 XML 문서 모델링을 위한 가장 기본적인 지식 중의 하나다. DTD란 SGML 또는 XML 문서의 구조와 내용을 정의하기 위한 것으로, 문서 안에서 쓰이는 엘리먼트(Element), 애트리뷰트(Attribute), 엔티티(Entity)의 정의와 각 요소간의 관계를 알려준다. XML 데이터를 자동으로 검증(Validation)하고 다른 회사와 문서 교환을 위해 어떤 형식의 문서 구조를 공유하고 싶을 때 DTD가 필요하게 된다. 그러나 DTD는 XML에 대한 규칙을 정의하는 데 있어 XML과 다른 문법을 사용한다. 따라서 일관성이 없고, 데이터 타입에 제한적이기 때문에 사용자가 원하는 형태를 정의해서 사용할 수가 없다. 이에 비해 스키마는 사용자 정의 데이터 타입을 선언할 수 있고, 다양한 데이터 타입을 지원하며 엘리먼트의 내용을 고유하게 정의할 수 있기 때문에 DTD를 스키마로 변환을 한다.

3.1 변환 규칙

<!ELEMENT> pcdata (#PCDATA)	<element name = 'pcdata'> <complexType content = 'textOnly'> </complexType> </element>
<!ELEMENT> any ANY	<element name = 'any'> <complexType> <any minOccurs = '0' maxOccurs = 'unbounded' processContenes = 'skip' /> <anyAttribute processContents='skip'/> </complexType> </element>
<!ELEMENT> empty EMPTY	<element name = 'empty'> <complexType content = 'empty'> </complexType> </element>
<!ELEMENT> empty EMPTY> <!ATTLIST empty notequired CDATA #IMPLIED avalue CDATA #FIXED 'fixedval' anenum (val1| val2| val3) #IMPLIED required CDATA #REQUIRED>	<element name = 'empty'> <complexType content = 'textOnly'> <attribute name='nonrequired' type='string'/> <attribute name='avalue' type='string' use='fixed' value='fixedval'/> <attribute name='anenum'> <simpleType base='NMTOKEN"> <enumeration value = 'val1'/> <enumeration value = 'val2'/> <enumeration value = 'val3'/> </simpleType> </attribute> <attribute name='required' type='string' use='required'/> </complexType> </element>
<!ELEMENT content (a , b)>	<element name = 'content'> <complexType content = 'elementOnly'> <sequence> <element ref = 'a' /> <element ref = 'b' /> </sequence> </complexType> </element>
<!ATTLIST Content a CDATA #REQUIRED>	<element name ='Content'> <complexType content="elementOnly". <attribute name='a' type='string' use='required'/> </complexType> </element>
<!ATTLIST Content a CDATA #IMPLIED>	<element name='content'> <complexType content='elementOnly'> <attribute name='a' type='string' use='optional'/> </complexType> </element>
<!ATTLIST Content a (x|y|z) #REQUIRED>	<element name='ROOT'> <complexType content='elementOnly'> <attribute name='a'> <simpleType base='string'> <enumeration value='x'/> <enumeration value='y'/> <enumeration value='z'/> </simpleType> </attribute> </complexType> </element>
<!ATTLIST Content a CDATA #FIXED "x">	<element name='content'> <complexType content='elementOnly'> <attribute name='a' type='string' use='fixed' value='x'/> </complexType> </element>
<!ATTLIST emp<!ATTLIST empty notreuired CDATA #IMPLIED avalue CDATA #FIXED 'fixedval' anunum (val1| val2| val3) #IMPLIED required CDATA #REQUIRED>ty notreuired CDATA #IMPLIED avalue CDATA #FIXED 'fixedval' anunum (val1| val2| val3) #IMPLIED required CDATA #REQUIRED>	<element name='empty> <complexType content='textOnly'> <attribute name='nonrequired' type='string' use='fixed value='fixedval'/> <attribute name='anenum'> <simpleType base='NMTOKEN'> <enumeration value = 'val1'/> <enumeration value = 'val2'/> <enumeration value = 'val3'/> </simpleType> </attribute> <attribute name = 'required' type='string use='required'/> </complexType> </element>

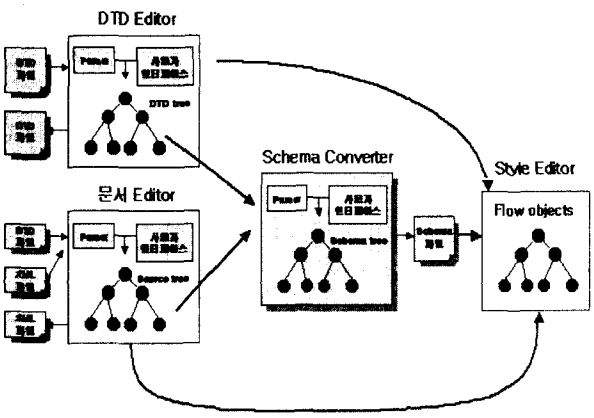
[그림 1] 변환 규칙

4. 변환기 설계 및 변환 과정

4.1 시스템의 구성

본 논문에서 설계한 XML 스키마 자동 변환기의 전체적인 시스템 구조는 아래의 [그림 2]와 같다. 사용자는 사용자 인터페이스를 통해서 새로운 DTD 파일을 작성하거나 기존의 DTD 파일을 열어서 편집

을 할 수가 있고 DTD 파일을 스키마로 변환 할 수가 있다. DTD 에디터에서 제공하는 사용자 인터페이스는 사용자가 DTD를 편집하기 쉽도록 해주며 문법에 맞는 DTD를 작성하도록 도와준다. 편집되는 DTD는 사용자 인터페이스와 XML 파서의 상호작용을 통해서 DTD 트리를 구성하게 된다. 또한 스키마 Converter를 통해 DTD 파일은 스키마로 변환이 되며, 사용자 인터페이스와 파서의 상호작용을 통해서 스키마 트리를 구성하게 된다. 사용자는 작성된 DTD/스키마에 스타일 에디터를 적용하여 각 엘리먼트에 대한 스타일을 지정할 수 있으며 그 결과를 스타일시트 파일로 저장할 수 있다.

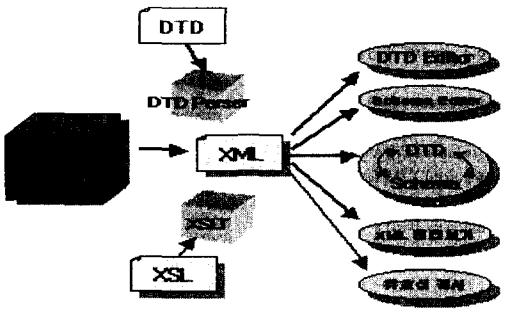


[그림 2] 시스템 구성도

4.2 시스템의 처리 요소

4.2.1 스키마 변환기

스키마 변환기는 DTD 파일과 XML 파일을 받아들여 DTD Editor와 문서 Editor로 각각 편집을 한 뒤 Schema Converter로 스키마 파일로 변환을 시켜 준다.



[그림 3] 스키마 Converter의 구조도

◆ Schema Parser

만들어진 XML 파일을 읽어들여서 [그림 1]의 변환 규칙을 이용하여 Schema Parser를 만들어서 DTD문서나 XML 파일을 문서에 구조를 부여하여 사용자가 작업할 수 있는 형태의 TREE구조로 변환되어 보여준다.

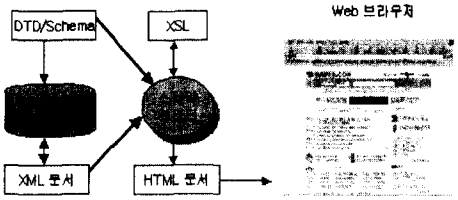
◆ DTD Parser

사용자가 XML 작업시 필요한 DTD 문서를 분석하여 보다 쉽게 DTD를 이해할 수 있도록 DTD에 대한 각종정보와 문서를 구조화한 형태의 TREE를 만들어 보여 준다.

◆ XSLT

XML과 XSL을 입력으로 받아 HTML로 변환하여 웹에서 보여질 수 있게 변환해 준다.

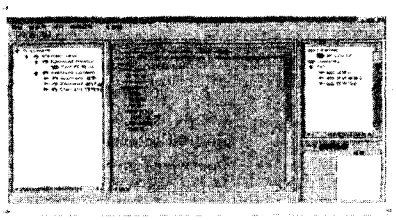
4.2.2 스타일 에디터



[그림 4] 스타일 에디터의 구조도

스타일 에디터는 DTD와 Schema에 따라 필요한 XSL 문서를 작성할 수 있도록 한다. 먼저 DTD/Schema 파일을 입력받아 Xvert를 스키마 자동 변환기를 통해 문서를 변환을 해주고 스타일 에디터를 이용하여 HTML 문서로 변환한 것을 Web 브라우저로 보여준다.

4.3 사용자 인터페이스



[그림 5] Xvert의 사용자 인터페이스

[그림 5]는 스키마 자동 변환기의 사용자 인터페이스를 나타낸다. 위의 인터페이스는 Visual C++을 사용하여 구현하였으며, 왼쪽에는 파일의 트리 구조를 나타내고, 가운데는 DTD/Schema 파일의 소스를 보여주며, 오른쪽은 DTD/Schema 파일의 트리구조를 보여준다.

5. 결론 및 향후 과제

현재까지 나와 있는 DTD를 스키마로 변환하는 변환기를 보면 아직까지 일반 사용자들이 편하게 사용할 수 있을 정도의 편리함을 가지지 못하고 있고, 변환 된 스키마를 재사용 할 수가 없다. 본 논문에서는 사용자들이 쉽게 접할 있는 사용자 인터페이스를 설계하였고, 스키마의 장점인 재사용이 가능하도록 하고, 모듈화가 될 수 있도록 제안을 하였다. 따라서 XML 문서와 DTD파일을 받아들여 스키마 파일로 쉽게 변환이 가능하다. 향후 연구과제로는 본 논문에서 제안한 변환기를 완벽하게 구현하고, 지정되어져 있는 XML 문서를 대상으로 하여 질의 시스템과 연계하는 것이다.

참고문헌

1.Frank Boumphrey 외 11인 저, "Professional XML Applications", 정보문화사, 2000

2.An Introduction to the Extensible Markup Language(XML), Matr in Bryan, http://www.personal.u-net.com

3."XSL Transformations (XSLT) Version 1.0 " http://www.w3.org/TR/xslt

4.박병진, "XML 기반의 통합 에디터 구현", 서강대학교 대학원 , 1999

5.Didier Martin외 12인 저, *Professional XML*, *Wrox*, 2000

6.Turbo XML v2.2 User Guide, 2001

7.XML Spy 3.0 User Guide, Icon Information Systems, 2001