제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

Schema 기반 XML 편집기 설계

박천수*, 강상승*, 한우용*, 손주찬*
*전자통신연구원 비즈니스 지식처리 연구팀
e-mail : bettle@etri.re.kr

Design of Schema-Based XML Editor

Cheon-Shu Park*, Sang-Seung Kang*, Woo-Yong Han*, Joo-Chan Son*
*Business Knowledge Research Team, ETRI

요 약

다양한 종류의 데이터 형태를 지원하기 위한 기존의 DTD 표현의 한계로 인해 W3C 에서 Schema 표준을 제정하였다. 최근 XML 의 적용분야가 다양해짐에 따라 애플리케이션에 맞는 데이터를 표현해야 되는 필요성을 갖게 되었다. 특히, e-Business 환경에서 구매요구서, 송장등과 같은 문서의 교환시 필요한 데이터에서는 좀더 세밀한 표현 형식을 요구하게 되었다. 본 논문에서는 다양한 형태의 데이터 표현이 가능한 스키마를 기반으로 XML 문서를 편집할 수 있는 편집기를 설계 하였다. 또한, 스키마를 받아들여 의미(semantics)를 파악하고 처리할 수 있는 모델을 제시하였다.

1. 서론

XML 은 자기 기술적인 데이터 형식으로 설계되어 저작자들이 문서의 내용을 기술하는 요소와 속성들의 이름을 정의하여 사용할 수 있다. W3C 는 XML1.0 을 권고안으로 채택할 때, 한 종류의 XML 문서들에 허용되는 구성요소를 제한하는 메커니즘으로 DTD(document Type Definitions)을 사용하였다.[4]

하지만, 최근 XML 의 적용분야가 기존의 문서 타입에서 벗어나 전자상거래, e-Business, 데이터 베이스를 비롯한 다양한 애플리케이션에서 처리하는 데이터들을 전송하고 표현해야 되는 필요성을 갖게 되었다. 따라서, XML 문서가 정적인 성격을 갖기 보다는 애플리케이션의 데이터로서 기능을 하기 위해, 기존의 DTD 에서 제공할 수 없는 세밀한 형태의 데이터 타입을 요구하게 되었다. 이러한 요구를 DTD 가 수용할 수 있는 한계가 있어 XML 스키마가 나오게 되었다. [1][2][3]

본 논문에서는, 현재 UN/CEFACT 와 OASIS 에서 진행되고 있는 ebXML(Electronic Business using XML)[5]의 핵심 컴포넌트(Core Component)[6]를 기반으로 작성된 스키마를 이용하여 새로운 비즈니스 문서를 생성하고 편집할 수 있는 스키마 기반의 편집기를 설계하였다. 본 시스템에서는 스키마를 입력으로 하여 새로운 XML 템플리트 인트턴스를 생성할 수 있는 모델을 제시하였다.

본 논문의 구성은 다음과 같다. 2 장에서는 스키마의 필요성과 배경에 대한 내용을 다루고, 3 장에서는 편집기 시스템의 전체적인 구조, 스키마의 의미 분석을 위한 세부 처리 모듈, GUI 와 XML 데이터간의 효율적인 처리를 위한 모델에 대해 설명한다. 4 장에서는 관련 연구를 살펴보고, 5 장에서는 결론 및 향후 연구 방향에 대해 설명한다.

2. 배경

2.1 스키마의 필요성

XML 문서에 대한 활용이 제한적인 환경에서는 기존의 문서에 대한 구조와 데이터 형태를 제공하기 위한 방법인 DTD 의 사용만으로 충분한 데이터 전송과 처리가 가능했었다. 그러나, DTD 의 문법은 점차 XML 문서의 활용이 다양해지고 새로운 분야에 쓰임에 따라서 XML 을 사용하는 사람들의 요구사항을 만족시키지 못했다. 또한, DTD 는 XML 문서와 전혀 다른 문법을 쓰기 때문에 문서를 핸들링 할 수 있는 또 다른 형태의 처리모델이 필요하게 되었다. 이러한 DTD 의 한계로 인해 W3C 에서는 새로운 형태의 문서구조와 데이터 형태를 제공하는 XML 스키마를 제정하였다.

XML 스키마는 XML 과 동일한 문법을 사용하므로 XML 처리 모델인 DOM(Document Object Model)을 이용하여 핸들링이 가능하게 되었고, 기존의 SQL, Java, VisualBasic, C++와 같은 언어들에서 볼 수 있는 원시 데이터형을 지원함으로써 좀더 세밀한 표현을 할 있고, 객체지향의 특성인 상속과 재활용을 증가 시켜 좀더 효율적인 문서 관리 및 처리가 가능하게 되었다.

DTD 와 XML 스키마는 문서에 대한 구조와 형태를 위한 기본 뼈대를 제공하는 메커니즘으로서 동일한 목적을 가지고 있으나 응용 분야와 목적에 따라서 전혀 다른 효과를 가질 수 있다. DTD 와 XML 스키마에 대해 좀더 자세히 살펴보면, 다음과 같은 차이점과 장단점을 알 수 있다.

우선, DTD 가 지원 가능한 데이터 타입은 CDATA, ID, NMTOKENS 등과 같은 String 형태로 제한적이다. 반면, XML 스키마는 기존의 프로그래밍 언어 수준에 가까운 19 가지의 원시 데이터형을 지원하고, 원시 데이터 형으로부터 유도 되어 44 가지의 단순형식(simple type)을 제공한다. 따라서 문서 내용을 애플리케이션이 요구하는 적절한 형식으로 제한하거나 확장하는 것이 가능하게 되었다. 이러한 데이터 타입을 이용하면 유지 보수를 쉽게 할 수 있을 뿐만 아니라 객체지향의 개념에서의 장점을 포함할 수도 있다.

둘째, XML 스키마는 XML 문법으로 작성되므로 새로운 형태의 문서가 아니며, XML 문서를 다룰 때 사용하는 표준들(DOM, SAX, XSLT, NameSpace 등)을 그대로 사용할 수 있다. 따라서 DTD 를 다룰 때 필요한 또 다른 형태의 처리 모델이 필요하지 않다.

셋째, XML 스키마는 다양한 형태의 내용모델(content model)을 제공한다. 즉, 이미 정의된 내용모델을 확장하거나 제한하는 것이 가능하며 재활용 할 수 있는 명시적인 방법을 제공한다. 또한, DTD 는 요소의 순서나 출현 횟수의 지정이 제한적인 반면, XML 스키마는 출현 횟수를 명시적인 값으로 줄 수 있으며, 그룹 개념을 지원하여 보다 강력한 내용 모델을 지원할 수 있고 요소와 그것의 내용을 표현하기 위한 융통성 있는 내용 모델을 제공한다.

마지막으로 XML 스키마는 문서 내의 Namespace 를 통해 필요한 만큼 여러 문서구조들을 가져와 하나의 문서내에서 여러 개의 스키마들을 참조할 수 있어, 문서에 대한 유효성 검증을 가능하게 하는 기능을 제공한다. 하지만, DTD 는 한 문서에 오직 하나의 DTD 만이 적용되며 XML Namespace 를 이용 하여 여러 개의 마크업들을 가져올 수 없다. 이처럼, XML 스키마는 ebXML, 전자상거래와 같은 다양한 종류의 데이터 형태를 공유하고 재상용 하는 비즈니스 문서의 사용에 응용 될 수 있다.

2.2 스키마의 구성 요소

XML 스키마는 기본 구성요소와 부가 구성요소로 나눌 수 있다.

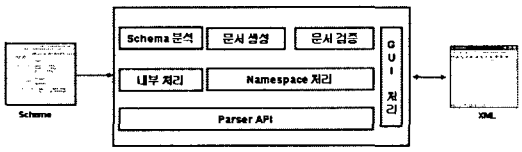
먼저, 기본구성요소로는 단순형식정의(simple type definitions), 복잡형식정의(complex type definitions), 요소선언(element declarations), 속성선언(attribute declarations)으로 이루어 진다. 단순형식정의는 자식요소를 포함하지 않는 단순 문자열로 string, boolean, decimal, float 등과 같은 원시 데이터형으로 구성된다. 복잡형식정의는 자식으로 다른 요소나 속성을 포함할 수 있는 복합형식이다. 요소 선언은 속성으로 이름(name)과 이름이 가질 수 있는 형태(type)을 가진다. 속성 선언은 속성 이름, 형태, 사용의 선택으로 이루어 진다.

부가 구성요소로는 include, redefine, import, annotation, compositor(sequence, choice, all), attribute use, wild card(skip, strict, lax), particles(minOccurs, maxOccurs), unique 등으로 이루어 진다. Include 는 동일한 namespace 에 속하거나 namespace 를 가지지 않는 스키마를 가져올 때 사용하고, redefine 은 이미 정의된 스키마에 대한 모델을 재정의 할 때 사용하며, import 는 다른 namespace 에 속해있는 스키마를 가져올 경우 사용한다. Annotation 은 어플리케이션에서 사용할 수 있는 스키마에 대한 설명을 포함한다. Sequence, Choice, All 은 요소에 대한 내용 모델을 제공한다. Attribute use 는 속성대한 제약을 줄 수 있다. Skip, Strict, Lax 는 프로세서가 이 요소 안에 있는 내용의 유효성 검증의 가부를 결정 할 때 사용한다. Particles 는 요소의 출현 횟수를 지정한다. Unique 요소는 한 문서 안에 있는 요소들이 고유한 값을 가지도록 규정할 수 있다.

3. 시스템 설계

3.1 시스템 구조

[그림 1]은 스키마 기반의 XML 문서를 생성하기 위한 시스템 구조를 나타내고 있다. 본 시스템은 크게 입력 받은 스키마를 분석하는 모듈, 분석된 스키마를 기반으로 의미를 파악하여 구조에 대한 관계를 처리하는 모듈, 문서를 생성, 검증 하는 모듈, 네임스페이스를 처리하는 모듈과 GUI 와 인터페이스를 담당하는 모듈로 이루어 진다.



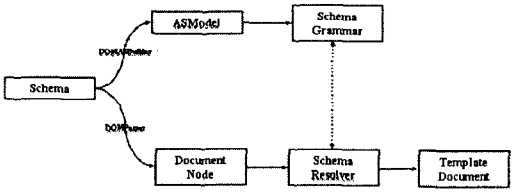
[그림 1] 스키마 기반 XML 편집기의 구조

스키마를 받아들여 DOM 프로세서를 이용하여 문서를 파싱 한 후, 스키마를 기반으로 한 XML 템플리트 인스턴스를 생성하기 위하여 최 상위 요소로 올수 있는 전역 요소들을 추출하여 사용자 인터페이스로부터의 선택된 최 상위 요소를 생성한다. 내부처리 모듈

을 통해 선택된 상위 요소의 자식으로 올 수 있는 하위 요소들에 대한 분석과 각각의 요소에 대한 타입, 속성 등을 분석하여 문서 생성모듈에 넘겨주게 된다. 문서 생성모듈은 각 요소들 간의 구조적인 관계와 정보를 이용하여 기본적으로 구성할 수 있는 XML 템플리트 문서를 생성하게 된다. 문서 생성시 *import* 되거나 include 되는 외부 스키마의 구조를 적용할 수 있도록 Namespace 를 처리하여 하나의 문서 내에 여러 개의 스키마를 사용하여 처리하도록 한다. 생성된 템플리트 문서에 대해서는 문서 검증을 통해 유효성을 검사하게 된다. 템플리트 문서는 사용자 인터페이스를 통해 사용자가 원하는 형태의 XML 인스턴스를 생성하게 된다. 최종적으로 생성된 문서는 스키마에 적합한 문서인지를 검사한다.

3.2 스키마 처리 모델

[그림 2]는 스키마를 처리하기 위한 모델을 나타낸다. W3C 에서 제공하는 스키마 문법(Schema Grammar)과 의미를 분석하여 계층적 구조에 대한 관계 정보를 유지하는 Schema Resolver 를 통해 템플릿 문서를 생성한다.

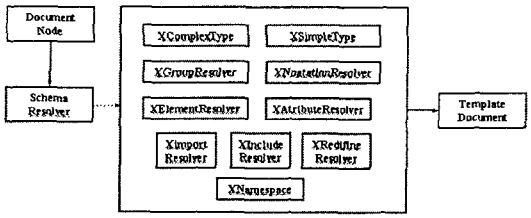


[그림 2] 스키마 처리 모델

입력 받은 스키마를 파싱하여 문서 노드(Document Node)와 ASModel(Abstract Schema Model)을 통한 스키마 문법을 얻어낸다. 스키마 문법에는 하나의 Namespace 내에 선언된 모든 스키마 컴포넌트를 포함하고 있다. 스키마 문법과 문서 노드를 이용하여 Schema Resolve 에서는 각각의 노드에 대한 구조적인 정보를 분석하고, 해석하여 템플리트 문서에 기본적으로 필요한 정보를 제공한다.

[그림 3]과 같이 Schema Resolver 는 문서를 파싱하여 얻은 document 노드를 입력으로 받아서 스키마의 구성요소에 맞는 모듈을 통해 의미와 관계를 분석하여 처리한다. 스키마의 구성요소는 2.2 절에서도 언급한 것처럼 기본 구성요소와 그 외의 부가 구성요소로 이루어 진다.

기본 구성 요소는 simple type, complex type, element declaration, attribute declaration 으로 이루어 진다. 부가 구성요소는 attribute group, identity constraint, model group, notation, annotation, compositor, wildcard, particles, include, import, redefine, unique 등으로 기본 요소 이외의 스키마를 .구성하는 요소이다.



[그림 3] Schema Resolver 의 구조

Schema Resolver 는 DOM API 를 이용하여 각 구성요소에 필요한 정보를 해석하고 템플릿 문서를 생성하기 위한 정보를 제공한다.

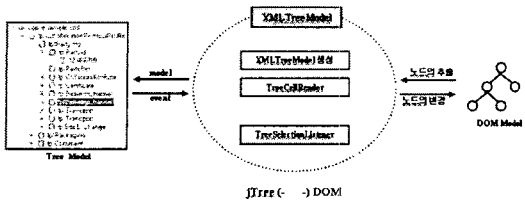
XComplexType 과 XSimpleType 은 요소에서 사용되는 형태를 해석하는 모듈이다. XImportResolver, XIncludeResolver, XRedefineResolver 는 외부 스키마에 대한 처리와 이미 정의된 스키마 모델을 제정의 하기 위한 모듈이고, XGroupResolver 는 Group 에 관련된 해석을 제공하는 모듈이다. XElementReslover 와 XAttributeResolver 는 각각 요소와 속성에 대한 해석을 처리하는 모듈이다. XNotationResolver 는 notation 에 대한 처리를 담당한다.

3.3 사용자 인터페이스(GUI)와 DOM 처리 모델

[그림 4]는 사용자 인터페이스와 DOM 의 처리를 위한 XML Tree 모델을 나타낸다. 본 시스템은 자바 환경에서 구현을 목적으로 하였다. 이에 사용자 인터페이스와 XML 문서에 대한 내부 처리 모델인 DOM 간의 효율적인 인터페이스가 필요하다. DOM 상에서 존재하는 모델을 사용자 인터페이스에서 표현하고 사용자 인터페이스 상에서 발생하는 이벤트를 처리하기 위해서는 여러 가지 방법이 있다.

우선, DOM 문서를 자바 환경의 JTree 에 메핑 하기 위해서는 각 노드를 방문하면서 JTree 에 맞는 XML Tree 모델을 생성해 주어야 한다. 생성된 XML Tree 모델을 이용하여 DOM 모델을 JTree 상에 나타내고, 사용자 인터페이스에서 발생한 이벤트 처리는 JTree 상에서의 XPath 표현을 추출하여 XPath 프로세서를 이용하여 처리하는 방법이 있다. 그러나, 이 방법은 이벤트가 발생할 때 마다 XPath 프로세서를 통해 실행한 결과를 가지고 노드의 변경을 해야 하므로 메모리의 낭비를 초래할 수 있다.

이에 본 시스템에서는 사용자 인터페이스와 DOM 이 동시에 사용할 수 있는 XML Tree 모델을 설계하였다. DOM 을 방문하면서 XML Tree 모델을 생성시에 DOM 노드 객체를 하나의 JTree 에서의 셀(cell)로 만들어 이벤트 발생시 선택된 JTree 컴포넌트 오브젝트를 XML 노드로 캐스팅 하여 직접 노드의 변경을 할 수 있도록 하여 JTree 와 DOM 을 동기화 하였다.



[그림 4] XML Tree 모델

4. 관련 연구

현재, XML 스키마 보다는 기존의 DTD 를 이용한 문서의 교환이 주를 이루고 있고, DTD 를 기반으로 한 상용 편집기도 많이 접할 수 있다. 하지만, 다양한 형태의 데이터를 전송하고 표현하기 위해 스키마에 대한 필요성이 점차 늘어나고 있다. 특히, ebXML 등과 같은 표준에서는 거래를 위한 비즈니스 문서의 형태를 스키마로 표현하고 있다.

Altova 에서 개발한 XML Spy[7]는 현재 상용제품으로 나와 있는 것 중 스키마 기반의 XML 문서 편집이 가능한 제품이다. XML Spy 가 일반적인 XML 문서를 핸들링 하기 위한 모델인데 반해, 본 시스템은 ebXML 환경에서 거래 데이터를 위한 스키마 모델에 대한 접근, 편집, 생성 할 수 있는 기능을 제공한다. 자바 환경에서의 스키마 기반 XML 편집기는 국내에서는 보기 힘들다. 국외에서는 상용 제품으로 Tiboco 에서 개발한 TurboXML[8]이 나와있으며, SUN 에서는 현재 브라우징 기능을 제공하는 상태로 진행 중에 있다.

5. 결론

최근 XML 의 적용분야가 다양해 지고, 기존의 문서 타입에서 벗어나 ebXML, 전자상거래등과 같이 데이터의 전송과 표현에 있어 좀더 복잡하고 세밀한 형태를 요구하는 어플리케이션에서 스키마의 필요성이 점차 증가하고 있다. 본 논문에서는, 현재 UN/CEFACT 와 OASIS 에서 진행되고 있는 ebXML 의 핵심 컴포넌트를 기반으로 작성된 스키마를 이용하여 새로운 비즈니스 문서를 생성하고 편집할 수 있는 스키마 기반의 편집기를 설계하였다. 또한 스키마를 받아들여 처리할 수 있는 모델을 제시하였다.

현재, 스키마의 의미를 파악하고 처리할 수 있는 전용 파서가 없으므로 기존의 DOM API 를 이용하여 스키마를 분석 한 후, 의미를 파악하기 위한 모듈을 따로 작성해야 한다. 향후, DTD 를 위한 전용 파서와 같은 XML 스키마를 위한 API 가 필요할 것이다.

참고문헌

[1] W3C, XML Schema Part 0:Primer, Recommendation, May. 2001.(http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/)

[2] W3C, XML Schema Part 1:Structures, Recommendation, May. 2001.(http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)

[3] W3C, XML Schema Part 2:Datatypes, Recommendation, May. 2001.(http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)

[4] Jon Duckett, Stephen Mohr, Oliver Griffin and Francis Norton, "Professional XML Schemas", 2001.

[5] ebXML, www.ebxml.org, UN/CEFACT and Oasis.

[6] ebTWG Core Components, Core Components Technical Specification Part I 1.8, UN/CEFACT, Feb. 2002.

[7] XML Spy, www.xmlspy.com, Alvota.

[8] Turbo XML, www.tibco.com, Tibco.