XML문서에서 UML 클래스 다이어그램 자동 변환

차남정*, 민미경*, 이숙희**
*서경대학교 컴퓨터과학과
**서경대학교 인터넷정보학과

Automatic Conversion of XML Documents to UML Class Diagram

Nam-Jeong Cha*, Mee-Kyung Min*, Sook Hee Lee**
*Dept. of computer Science, Seokyeong Univ.
**Dept. of Internet Information, Seokyeong Univ.

요 약

XML 문서의 구조를 파악하기 위하여 현재 많은 연구가 진행 되고 있으며, 대부분 XML 문서의 구조를 찾아 DTD나 스키마로 표현하는데 중점을 두고 있다. 본 논문에서는 XML 문서에서 구조를 추출하여 이를 UML 클래스 다이어그램으로 자동 변환하는 시스템을 제안한다. 제안된 시스템에서는 XML 문서로부터 요소-속성 트리를 구성하고, 이를 활용하여 문서 구조를 UML 클래스 다이어그램으로 쉽게 변환하도록 한다.

1. 서론

XML(eXtensible Markup Language)[1]의 사용범위가 확대됨에 따라 인터넷을 통해서 많은 양의 XML 문서가 교환되어 축적되고 있다. 이 축적된 문서를 활용하기 위해서 XML 문서의 구조를 파악하는 것이 중요하다. XML 문서의 구조를 DTD(Document Type Definition)나 스키마를 이용해서 파악하는 것이 이해하기 쉽고, 간편하며, 정확하다. 하지만, DTD나 스키마는 XML 문서 내부에 포함되어 있는 경우가 거의 없으며, 외부에서 접근이 어렵다. 따라서 본 논문에서는 DTD, 스키마가 존재하지 않아 구조 파악이 어려운 문서의 구조를 파악하여, 그 결과를 UML(Unified Modeling Language)[3]로 표현하는 방법을 제안하고자 한다. UML은 OMG에서 표준으로 인정한 객체지향 분석, 설계를 위한 모델링 언어이다. UML은 사용자가 이해하기 쉬운 시각적 언어로 시스템을 표현하므로 복잡한 구조를 설명하는데 적합하다. XML 문서의 구조를 설명하는 별도의 문서(DTD, 스키마)가 존재하지 않아서 새로 문서를 작성함으로써 역공학을 이용할 수 있다. 역공학이란 이미 만들어진 시스템을 역으로 추적하여 처음의 문서나 설계기법 등 자료를 얻어내는 것을 말한다.

본 논문에서는 XML 문서 구조 파악을 위하여 요소-속성 트리를 제안한다. 요소-속성 트리는 XML 문서의 구조를 간략하게 표현하여 UML 클래스 다이어그램으로 쉽게 변환하기 위해 고안된 트리이며, XML 문서의 요소, 속성 모두를 트리의 노드로 반복 없이 표현 한다. 또한 XML문서의 구조를 UML로 사상시키기 위해 UML과 XML사이에 발생하는 이질성 보완을 제안 하였으며, XML 문서의 구조를 UML로 사상 시키는 도중에 발생 가능한 상황에 대해 설명하고 관련 규칙을 정의하였다.

다음 2장에서 관련연구를 다루며, 3장에서는 요소-노드 트리생성의 정의와 생성 방법에 대하여 설명한다. 4장에서 UML 클래스 다이어그램으로의 변환규칙을 정의, 제안한다 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

본 장에서는 기존 XML 문서에서 구조를 파악하는데 보편화 되어 있는 스키마 추출법에 대하여 서술 하고자 한다.

스카마 추출은 추출방법에 따라 「트리 표현식의 발생빈도에 따른 스키마 추출방법」과 「발생빈도 패턴에 따른 스키마 추출방법」으로 나눌 수 있다[5].

「트리 표현식의 발생빈도에 따른 스키마 추출방법」이란, 먼저 데이터를 「트리 표현식」으로 표현한 후, 표현된 「트리 표현식」중에서 사용자가 정한 지지도 보다 높으며, 많은 정보를 표현하고 있는 「트리 표현식」을 스키마로 정의하는 방법이다. 자주 발생되는 비슷한 질의에 대해 「트리 표현식」을 만들어 발생빈도에 따라 스키마를 추출함으로, 유사한 질의에 대하여 효과적으로 실행되는 장점이 있다. 단점으로는 문서 전체에 대한 스키마를 찾는 데는 어려움이 따른다.

「발생빈도 패턴에 따른 스키마 추출방법」이란

반복적으로 나타나는 빈도수를 바탕으로 스키마를 추출하는 방법이다. 이 방법은 사용자가 정한 발생 빈도수를 바탕으로 스키마가 다양하게 추출된다는 장점은 있지만 전체 스키마보다는 특정 패턴에 대한 스키마가 추출이 되는 제약이 있다.

그 외에 유명한 방법으로는 DataGuide와 Datalog를 이용한 방법이 있다. DataGuides는 주어진 데이터 그래프를 간결하며, 정확하게 요약하기 위해 사용한다. DataGuides는 모든 구조 정보로 구성된 최대경계스키마와 공통적인 구조만으로 구성된 최소경계스키마를 생성한다. 최대경계 스키마는 전체 문서에 대한 질의를 수행해야 하는 단점이 있으나 정확한 검색을 수행한다. 반대로 최소경계스키마는 질의의 범위를 최소화시키는 장점이 있다. Datalog는 규칙기반 언어로 스키마를 추출하기 위해서 최대고정점(maximum fixed point)을 이용하는 최소경계 스키마 추출방법의 일종이다. Datalog는 데이터에 존재하는 객체들이 최대고정점에 도달할 때까지 반복해서 접근하는 방법으로, 스키마 추출 시 많은 비용이 요구되며, 스키마 갱신시 부담이 증가한다[4].

3. 요소-속성 트리 생성

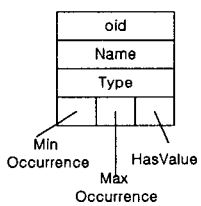
3.1 요소-속성 트리(Element-Attribute Tree)

3.1.1 요소-속성 트리의 정의

요소-속성 트리는 XML 문서를 이루는 요소와 속성을 트리의 노드로 표현하며, XML 문서의 구조 경로를 트리의 경로로써 표현한다. 이렇게 표현된 요소-속성 트리는 XML 문서의 구조를 잘 표현 한다는 장점을 가지게 된다. 요소-속성 트리의 노드는 노드의 타입 값이 'Element' 인 요소 노드와 'Attribute' 인 속성 노드로 구성된다. 요소 노드는 요소 노드와 속성 노드 모두를 자식 노드로 가질 수 있으며, 속성 노드는 자식을 가지지 못한다.

3.1.2 노드의 구조

요소-속성 트리 노드의 구조는 [그림 1]과 같다. oid는 노드의 ID를 저장하며, Key의 역할을 한다. Name은 요소나 속성의 명칭을 저장하기 위한 부분으로 요소, 속성의 명칭을 그대로 저장한다. 예외적으로 요소가 NameSpace로 한정 되어 있는 경우 NameSpace 접두사와 요소의 명칭을 ':' 을 이용해 연결한 문자열을 저장한다. Type은 노드의 타입으로 'Element' 또는 'Attribute' 값이 저장된다. Min Occurrence, Max Occurrence는 요소나 속성이 나타나는 최소/최대 발생횟수이다. 마지막으로 HasValue는 요소나 속성이 값을 갖는지 여부를 저장한다.



[그림 1] 노드의 구조

3.1.3 요소-속성 트리 생성방법

요소-속성 트리를 생성 하기 위해서 아래 순서를 따른다.

1. XML 문서의 루트 요소가 나올 때까지 XML문서를 이루는 다른 요소-속성은 무시한다.
2. XML 문서의 루트 요소를 요소-속성 트리의 루트 노드로 변환 시킨다.
3. 노드로 변환된 요소가 자식 요소를 포함하는지 확인한다.
   3.1. 자식 요소가 있는 경우 자식 요소가 이미 변환되어 요소-속성 트리에 있는지 확인한다.
      3.1.1 이미 변환 되어 있는 경우, 자식 노드로 추가 하지 않는다.
      3.1.2 변환 되어 있지 않은 경우, 자식 요소를 변환하여 자식 노드로 생성한다.
   3.2. 계속해서 자식 요소가 있는 경우는 깊이 우선 방식으로 3번을 반복한다.
4. 노드에 속성을 포함 하는지 확인한다.
   4.1 노드에 속성이 있는 경우 속성이 이미 변환되어 요소-속성 트리에 있는지 확인한다.
      4.1.1 이미 변환 되어 있는 경우, 자식 노드로 추가 하지 않는다.
      4.1.2 변환 되어 있지 않은 경우, 속성을 변환하여 자식 노드로 생성한다.
   4.2 계속해서 속성이 있는 경우 4.1을 반복한다.
5. 이웃한요소에 대하여 3번부터 반복한다.

[그림 2] 요소-속성 트리 생성방법

3.2 요소-속성 트리의 생성 예

3.2.1 XML 문서 예

아래 [그림 3]의 예제 XML 문서는 2명의 저자와 2권의 책 정보를 담고 있는 문서이며, Xlink와 ID, IDREF, IDREFS를 속성값으로 사용하고 있다.

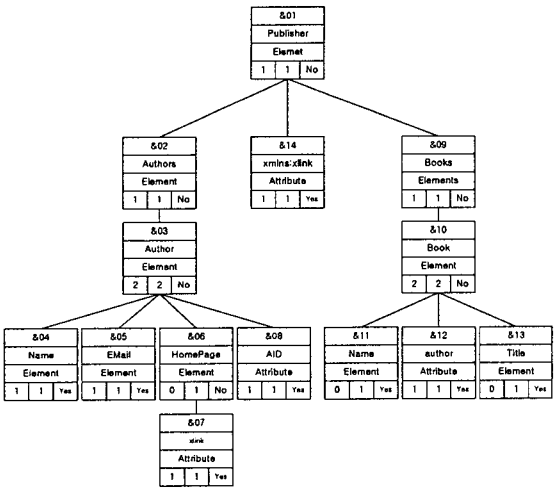
```
<?xml version="1.0"?>
<Publisher
xmlns:xlink="http://www.w3.org/1999/xlink/namespace/">
  <Authors>
    <Author AID="JHN_SM">
      <Name> Jhon Smith</Name>
      <EMail> JS@kura.com</EMail>
      <HomePage xlink:type="simple"
      xlink:href=http://www.naver.com
      xlink:actuate="onLoad" />
    </Author>
    <Author AID="MIC_JS">
      <Name> Michal Jackson</Name>
      <EMail> Mj@kura.com</EMail>
    </Author>
  </Authors>

  <Books>
    <Book author="JHN_SM">
      <Name>For Beginner</Name>
    </Book>
    <Book author="JHN_SM MIC_JS">
      <Title>About XML</Title>
    </Book>
  </Books>
</Publisher>
```

[그림 3] 예제 XML 문서

3.2.2 요소-속성 트리 예

아래 [그림 4]는 [그림 3]의 XML 문서를 요소-속성 트리로 표현한 것이다.



[그림 4] 요소-속성 트리

4. UML 클래스 다이어그램 변환

4.1 UML 클래스 다이어그램 변환규칙

요소-속성 트리를 클래스 다이어그램으로 변환하기 위해서는 클래스, 관계, 다중성에 대하여 각각 고려해야 한다.

4.1.1. 클래스

모든 요소 노드는 클래스로 변환 시킨다. 클래스의 명칭은 노드의 명칭을 사용하며, 타입은 Stereo Type인 <<ElementType>>으로 변환시킨다[2]. 예외적으로 요소의 명칭에 ' Xlink' 나 ' xmlns' 가 사용된 경우에는 각각 Stereo Type인 <<DataType>>의 Xlink, NameSpace 클래스로 변환시킨다. String 클래스는 HasValue가 True인 요소 노드가 있는 경우 자동 생성되며, 타입은 Stereo Type인 <<DataType>>으로 한다.

모든 속성 노드는 부모 노드인 요소 노드가 변환되어 생성된 클래스의 속성이 된다. 속성 노드의 명칭에 ' xlink' 또는 ' xmlns' 가 사용된 경우에는 요소의 명칭에 사용된 경우와 같이 변환 시킨다.

4.1.2 관계(Relationship)

UML 클래스 다이어그램에서 관계의 종류는 의존 관계(Dependency), 일반화 관계(Generalization), 연관 관계(Association)가 있으며, XML 문서의 구조를 클래스 다이어그램으로 표현 시에는 일반화 관계와 연관 관계만이 사용된다. 연관관계는 부모 요소 노드와 자식 요소 노드 사이의 관계를 표현할 때 사용된다. 부모 요소는 자식 요소로 구성되어 있으므로 연관 관계 중 복합 연관 관계로 변환시킨다. 일반화 관계는 Xlink, NameSpace, String 클래스와 요소 노드가 변환된 클래스간의 관계 표현에 사용된다. 요소나 속성 노드 명칭에 ' xlink:' 가 사용된 경우에는 사용된 노드의 부모 노드에 해당하는 클래스와 Xlink 클래스 사이에 일반화 관계가 생성된다.

요소나 속성 노드 명칭에 ' xmlns:' 가 사용되는 경우가 있다. 이 경우에는 사용된 노드의 부모 노드에 해당하는 클래스와 NameSpace 클래스 사이에는 일반화 관계가 생성된다. 또한, 부모 노드에 해당하는 클래스에서 NameSpace 클래스 방향으로 역할이 생성되며, 역할명칭은 NameSpace 접두사를 이용한다.

요소 노드의 HasValue가 ' True' 인 경우에는 요소 노드가 변환된 클래스와 String 클래스 사이에는 일반화 관계가 생성 된다. 요소 노드가 변환된 클래스에서 String 클래스 방향으로 역할이 생성되며, 역할의 명칭은 요소 노드의 명칭을 이용한다.

4.1.3 다중성(Multiplicity)

다중성을 생성하기 위해서는 요소-속성 트리에서 각 노드에 저장되어 있는 최소/최대 발생 값을 이용한다. 요소 노드에 저장된 값은 클래스 사이의 다중성을 나타내며, 속성 노드에 저장된 값은 속성 노드의 부모노드에 해당하는 클래스에 포함된 속성의 다중성을 나타낸다.

4.2 UML 클래스 다이어그램의 DTD 표현

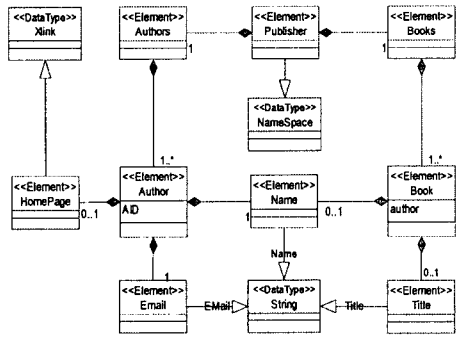
XML 문서의 구조를 파악하기 위한 UML을 생성하기 위해서는 요소-속성 트리를 이용하여, 아래의 DTD를 참조하는 XML 문서를 생성한다. 아래 [그림 5]에서 제안된 DTD는 XML 구조를 설명하기 위해 특성화된 UML을 위한 것이다.

```
<!ELEMENT ClassDiagram (Classes?, Relations?) >
    <!ATTLIST ClassDiagram
        Name CDATA #REQUIRED
    >
<!ELEMENT Classes (Class*) >
    <!ELEMENT Class (Attribute*) >
        <!ELEMENT Attribute EMPTY>
            <!ATTLIST Attribute
                Name CDATA #REQUIRED
                Type CDATA #FIXED "String"
            >
    <!ATTLIST Class
        Name CDATA #REQUIRED
        Type (DataType | ElementType) #IMPLIED
    >
<!-- Declare Relation -->
<!ELEMENT Relations (Relation*) >
    <!ELEMENT Relation (From, To)>
        <!ELEMENT From EMPTY>
            <!ATTLIST From
            Name CDATA #REQUIRED
            Role CDATA
            Multiplicity (Zero | One |
            ZToO | ZToMa | OToMa) #IMPLIED
            >
        <!ELEMENT To EMPTY>
            <!ATTLIST To
            Name CDATA #REQUIRED
            Role CDATA
            Multiplicity (Zero | One |
            ZToO | ZToMa | OToMa) #IMPLIED
            >
    <!ATTLIST Relation
    Name CDATA
    Type (General | Association | Dependency) #IMPLIED
    >
```

[그림 5] UML 표기를 위한 DTD

4.3 UML 클래스 다이어그램 생성 예

[그림 6]은 [그림 3]의 문서를 UML 클래스 다이어그램으로 표현한 것이다. [그림 3]의 예제 문서와 비교 하면, ' Author' 요소는 ' Author' 클래스로 변환 되어 있으며, 부모 요소인 ' Authors' 가 변환된 ' Authors' 클래스와 복합 연관으로 관계를 맺고 있다. 요소에서 값을 사용한 Name 클래스는 String 클래스와 일반화 관계를 맺고 있다.



[그림 6] UML로 변환된 XML

[그림 7]은 [그림 6]의 UML을 XML로 나타낸 형태로서 클래스 부분과 관계, 다중성 부분으로 나뉘어져 있다.

```
<ClassDiagram Name="Publisher">
  <Classes>
    <Class Name="Publisher" Type="Element" />
    <Class Name="Authors" Type="Element" />
    <Class Name="Author" Type="Element">
      <Attribute Name="AID" Type="String" />
    </Class>

     ... 중간 생략 ...
    <Class Name="HomePage" Type="Element"/>
    <Class Name="Name" Type="Element" />
    <Class Name="XLink" Type="DataType"/>
    <Class Name="String" Type="DataType"/>
  </Classes>

 <Relations>
    <Relation Type="Association">
      <From Name="Authors" Multiplicity="One" />
      <To Name="Publishers" />
    </Relation>
    <Relation Type="Association">
      <From Name="Author" Multiplicity="OToMa" />
      <To Name="Authors" />
    </Relation>
    <Relation Type="Association">
      <From Name="HomePage" Multiplicity="ZToO" />
      <To Name="Author" />
    </Relation>
```

[그림 7] 결과 UML

```
... 중간 생략 ...

    <Relation Type="General">
      <From Name="Email" Role="EMail"/>
      <To Name="String" />
    </Relation>
    <Relation Type="General">
      <From Name="Publishers" />
      <To Name="NameSpace" />
    </Relation>
    <Relation Type="General">
      <From Name="HomePage" />
      <To Name="XLink" />
    </Relation>
  </Relations>
</ClassDiagram>
```

[그림 7] 결과 UML(계속)

5. 결론

기존에 XML 문서의 구조를 파악하는 방법들은 파악된 구조를 DTD나 스키마로 변환한다. 그러나 DTD나 스키마는 사람이 읽고 이해하는 것을 목적으로 만들어진 문서가 아니기 때문에 문서의 구조를 이해하기 위하여 별도의 노력이 필요하다. 그러나 본 논문에서 제안하는 방법은 XML 문서에서 파악한 구조의 결과를 시각적 언어인 UML로 변환한다. 따라서 XML 문서 사용자는 직관적으로 문서의 구조를 이해할 수 있으며, 문서의 갱신 및 생성을 용이하게 할 수 있다는 장점을 갖는다.

본 논문에서는 XML 문서의 구조를 UML로 표현하기 위하여 요소-속성 트리의 사용을 제안하였다. 요소-속성 트리는 XML 문서의 구조를 간단하게 표현하며, 별도의 절차 없이 바로 UML로 변환 되기 때문에 매우 유용하다.

따라서 향후 과제로는 문서에 사용된 자료형을 파악 한 후 이를 UML 클래스 다이어그램 자동 변환에 이용하는 방법과 Xlink, XPath, XPointer가 함께 사용된 문서의 처리 방법에 관한 연구가 필요하다.

[참고문헌]

[1] W3C, " eXtensible Makeup Language(XML) 1.0 Second Edition" , *http://www.w3.org/TR/REC-xml*, Oct 2000.

[2] Grady Booch , Magnus Christerson, Matthew Fuchs and Jari Koistinen, " UML for XML Schema Mapping Specification" , *Http://www.rational.com/media/uml/resources/media/uml_xmlschema33.doc*, Dec 1999.

[3] James Rumbaugh, Ivar Jacobson and Grady Booch, *The Unified Modeling language Reference Manual*.

[4] S. Abiteboul, P. Bunneman and D. Suciu, " Data on the Web : From Relations to Semistructured Data and XML" *Morgan Kaufmann*, 1999.

[5] 김성림, 윤용익, " XML 문서에서의 엘리먼트 정보를 이용한 스키마 추출방법" , 한국정보처리학회 논문지 제9-D권 제3호, Jun 2002.