제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003.11)

UML 다이어그램의 XML 문서 변환에 관한 연구

이정석*, 박해우, 강병욱
영남대학교 컴퓨터공학과
e-mail:crossninja@yumail.ac.kr

A study on Conversion of UML Diagram to XML Documents

Jeong-Seok Lee*, Hea-Woo Park, Byung-Wook Kang
Dept. of Computer Engineering, Yeung-Nam University

요 약

XML(eXtensible Markup Language) 프로그램이나 XML을 이용한 B2B 시스템 구축과 같은 XML 프로젝트에서는 객체 지향적 설계언어인 UML을 이용해 개발하면 효율을 높일 수 있다. UML(Unified Modeling Language)로 XML문서 구조를 표현하는 이유는 XML문서를 생성, 접근, 수정하는 XML프로그램을 체계적이고 효율적으로 설계할 수 있기 때문이다. DTD(Document Type Declaration)와 스키마(Schema)를 UML로 표현함으로써 프로그래밍을 통합적으로 추진 할 수 있다. 이러한 과정에서 XML의 문서 구조정보의 활용 증대와 UML의 확장이라는 이점을 취할 수 있다. 본 논문에서는 UML 기반의 다이어그램에서 XML 문서로의 변환기에 대한 모델을 제안한다.

1. 서론

XML(eXtensible Markup Language)은 인터넷의 중간 조직적인 전달을 위한 메타문법으로서 신속하게 표준화되어 왔다. 그래서 비지니스 분석가, 시스템 분석가, 소프트웨어 개발자들은 XML로 나타낸 정보 모델을 만들고, XML과 그것을 프로세스하는 시스템 사이의 관계를 기술하는 것이 점점 필요하게 되었다[1].

UML은 교환 포맷을 활용하여 데이터 모델을 제공하기 위한 중요한 대안이지만 기존의 교환 포맷은 재공학을 목표로 하였기 때문에 메소드 호출과 변수 접근과 같은 내부 의존에 치중하여 생성 되었다. 따라서 포맷의 소스코드로부터 UML을 추출하는데 정확하고 직접적인 매핑 과정의 어려움이 따르는 특징이 있다. 하지만 XML은 어떠한 정보도 표현할 수 있는 범용적인 프레임웍(Framework)을 제공하기 때문에 UML 모델 정보를 XML의 구조적인 정보로 설계할 수 있다.

XML은 구조화된 정보를 포함하고 있는 문서들을 위한 마크업 언어이다. 구조화된 구체적인 내용과 그 내용이 수행해야 할 역할을 포함하고 있다.

XML은 객체 기반의 언어가 아니다. Tree 구조의 문서 형식에 친화적이기 때문에 객체 지향의 기본적인 개념들 다형성이나 다중상속, 복합 등과 같은 특징들을 기술하기에는 무리가 있다.

XMI(XML Metadata Interchange)란 UML을 위한 객체 모델의 표준이다. 즉, UML 객체 모델의 메타 요소들의 표준으로 UML 다이어그램간의 호환을 책임진다. 각각의 객체 모델의 정의는 XML스키마와 DTD에 정의되어 있으며, XML 스키마를 통해서도 정보의 손실 없이 UML의 요소를 생성할 수 있다.

XML이 새로운 문서 교환 표준으로 정착되고 있는 현실에서 UML로 XML문서 구조를 표현하는 이유는 XML문서를 생성, 접근, 수정하는 XML프로그램을 체계적이고 효율적으로 설계할 수 있기 때문이다. XML 프로그램이나 XML을 이용한 B2B 시스템 구축과 같은 XML 프로젝트에서는 객체 지향적 설

계언어인 UML을 이용해 개발한다면 효율을 높일 수 있을 것이다. 프로그래밍 할 때 개별적인 XML 문서 내용보다 중요한 것은 XML 문서의 구조정보를 얻고 이를 활용하는 것인데, DTD와 스키마가 바로 이런 역할을 한다. 그러므로 DTD와 스키마를 UML로 표현함으로써 프로그래밍을 통합적으로 추진 할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 XML과 UML에 관한 연구를 기술하였고, 3장에서는 XML을 이용한 UML 다이어그램의 표현 방법에 대한 정의 및 규칙을 기술하고, 4장에서는 UML 다이어그램의 XML 문서변환에 대한 설계에 대해 기술하고, 5장에서는 결론 및 향후 연구 방향에 대해서 기술하였다.

2. 관련 연구

본 논문의 키워드가 되는 UML과 XML의 적용 가능한 규칙들을 살펴본다.

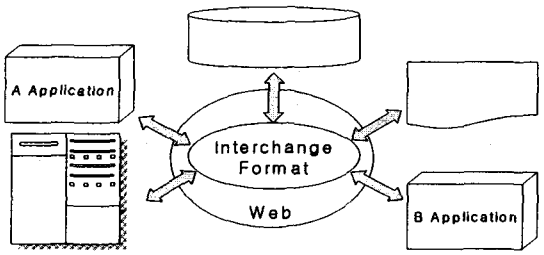
2.1 XML의 데이터적 의미

XML 어휘는 모든 조건을 만족시키며 모든 유형의 컨텐츠(Content)를 표현할 수 있도록 설계되었다. 이것은 특정한 유형의 컨텐츠를 표현하고자 할 때 선택할 수 있는 어플리케이션의 폭이 넓어진다는 의미이다[2]. 모든 데이터는 XML로 저장될 수 있으며 저장된 데이터의 검증은 특정 컨텐츠를 표현하는 태그와 속성을 정의한 DTD(Document Type Definition)를 통해 할 수 있다. 또한 저장된 XML 데이터는 다양한 XML 파서(Parser)를 사용해 쉽게 읽어들이고 변경할 수 있다[3]. 그래서 XML은 프로그램 작성이 용이하다는 장점도 제공하게 된다.

2.2 XML의 저장매체로서의 의미

XML은 고도로 복잡한 정보를 저장하기 위한 효율적이 저장매체로서의 기능을 가지고 있다. 만약 데이터들이 계층적 구조를 가지고 있다면 그 구조를 저장매체에도 동일하게 가져가는 것이 효율적이다[4]. XML은 메타데이터로서 스스로 해석 가능한 구조를 가지고 있어서 데이터 교환을 위한 공용 형식으로 사용될 수 있다[2][4]. XML을 데이터 교환 양식으로 사용하면 응용프로그램이 모든 데이터들을 하나의 논리적인 관점에서 다룰 수 있게 할 수 있는 장점을 제공해 준다.

그림 2.1은 XML이 데이터 교환양식으로서의 기능을 보여주고 있다.

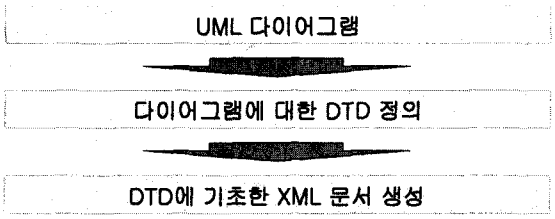


<그림 2.1> XML을 이용한 데이터 교환

2.3 UML 다이어그램의 표현

객체지향 모델을 문서화 하는데 필수적인 개념들과 표기법들의 대부분을 제공하는 UML은 현재 그 모델을 상호 교환하기 위한 명백한 포맷을 갖고 있지 않다. 따라서 UML 모델의 저장과 공유를 위해 웹 환경에서 응용프로그램에 중립적인 교환 포맷으로 XML을 이용하였다. DTD를 기초로 정의된 요소와 속성 태그에 대해 UML 정보를 표현함으로써 XML 문서를 생성한다.

XML을 이용한 UML 다이어그램의 표현은 그림 2.2와 같이 나타낼 수 있다.



<그림 2.2> XML을 이용한 UML의 표현

3. XML을 이용한 UML 다이어그램의 표현

본 장에서는 UML 다이어그램을 XML로의 변환에 관한 내용을 살펴본다.

3.1 XML의 구조

XML 문서를 설계할 때는 마크업(태그, 속성, 엔티티)을 사용하여 컨텐츠를 표현할 수 있다. DTD는 문서를 표현하는데 사용할 수 있는 마크업 요소를

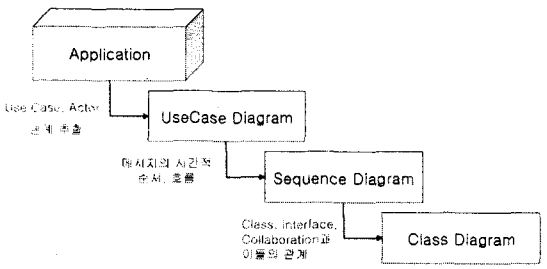
정의한 언어 규약집이라 할 수 있다[3]. XML의 구조는 간단히 DTD와 XML 문서 자체로 구성되어 있다. XML의 문서구조는 논리적 구조와 물리적 구조로 구성된다. 논리적 구조는 문서의 전체적인 구조를 표현하는 부분으로 요소, 속성 등의 구성요소를 이용하여 표현하며, 물리적 구조는 엔티티(Entity)를 이용하여 표현한다. 이들이 혼합되어 문서의 구조를 나타내는 DTD를 형성한다[4][5].

UML 모델에 대한 정보를 DTD에 정의하기 위해서는 모델을 표현할 수 있는 모든 요소들을 추출하여 DTD에 태그로 정의해 주어야 한다. 이를 바탕으로 실제 UML 모델에 대한 XML 표현이 가능하다.

3.2 UML 다이어그램의 한정

UML에서는 Class, Object, UseCase, Sequence, Collaboration, State Chart, Activity, Component, Deployment의 9가지 다이어그램을 제공하고 있다.

보편적 모델링 기법에 적용하면 모델에 따라 사용하는 다이어그램들의 차이가 있다[6]. 즉 어떠한 모델을 구성함에 있어서 UML에서 제공하는 다이어그램의 전부가 필요하지는 않다는 것이다. 단일 컴퓨터에서 실행되는 단순한 일체형의 어플리케이션을 모델링 하는 경우라면 UseCase 다이어그램, Sequence 다이어그램, Class 다이어그램만으로 충분한 표현이 가능하다. 본 논문에서는 단순한 일체형 모델의 UML 다이어그램 변환으로 설정하였다.

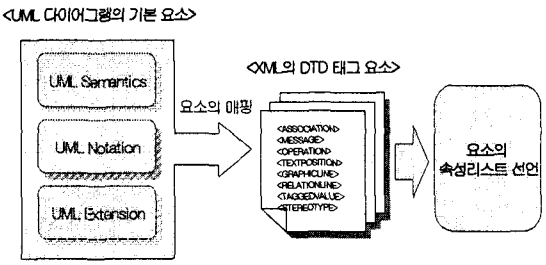


<그림 3.1> UML 다이어그램의 추출

그림 3.1은 어플리케이션 모델에서 UML 다이어그램을 추출하는 과정을 보여주고 있다.

3.3 UML 다이어그램의 DTD 정의

XML 문서에서 사용할 UML 다이어그램의 태그를 정의하고, 이들이 어떤 순서로 동작하며, 모델링 하고자 하는 객체들의 관계와 의미 정보를 포함하는 DTD를 우선적으로 설계해야 한다. DTD내에는 UML 메타모델의 모델 요소들을 갖고 있어야 하며 이러한 각 요소들은 실제로 XML문서의 태그로 정의된다[7]. 따라서 UML DTD는 UML 다이어그램의 모델 요소에 대한 속성들의 정의가 필요하다.



<그림 3.2> UML의 DTD 모델링 과정

그림 3.2는 UML의 요소를 XML DTD로 모델링 하는 과정을 보여주고 있다.

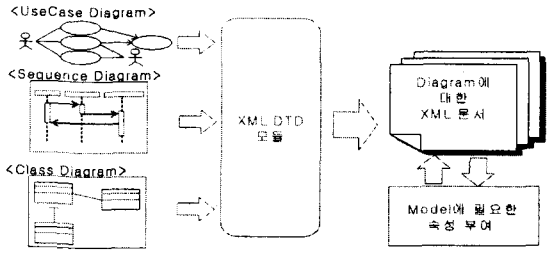
```
<!ELEMENT Class (Note*, Name, IsActive?, TemplateParameters?
                , %ClassifierFeature;)>
<!ELEMENT IsActive (#PCDATA)>
<!ELEMENT Templace (Note*, Name, %ClassifierFeature;)>
<!ELEMENT AssociationClass (Note, Name, AttachedAssociation,
                %ClassifierFeature;)>
<!ELEMENT AttachedAssociation (#PCDATA)>

<!ELEMENT Attribute (Note*, Name, DataType?, Visibility?, InitialValue?,
                TargetScope?, Changeable?, Multiplicity?)>
<!ELEMENT Visibility (#PCDATA)>
<!ELEMENT InitialValue (#PCDATA)>
<!ELEMENT TargetScope (#PCDATA)>
<!ELEMENT Changeable (#PCDATA)>
<!ELEMENT Multiplicity (#PCDATA)>

<!ELEMENT Operation (Note*, Name, Visibility?, ReturnType?, Concurrency?,
                IsAbstract? , IsRoot?, OwnerScope?, Parameter*)>
<!ELEMENT Parameter (Note*, Name, Type?, DefaultValue?, Kind?)>
<!ELEMENT ReturnType (#PCDATA)>
<!ELEMENT Concurrency (#PCDATA)>
<!ELEMENT IsAbstract (#PCDATA)>
<!ELEMENT IsLeaf (#PCDATA)>
<!ELEMENT IsRoot (#PCDATA)>
<!ELEMENT OwnerScope (#PCDATA)>
```

<그림 3.3> 클래스에 대한 XML DTD

그림 3.3은 그림 3.2의 모델링 과정을 거쳐 정의한 클래스의 DTD를 보여준다.

4. UML 다이어그램의 XML 변환

3장에서 정의한 DTD를 모듈로 하여 UML 다이어그램에서 XML 문서를 자동 변환하는 UML-to-XML 변환기를 설계 하였다.
어플리케이션 모델의 설계시 요소를 추출하여 UML 다이어그램을 작성할 수 있다. 작성된 UML 다이어그램에서 그림 4.1의 UML-to-XML 변환기를 통해 XML 문서화 한다.



<그림 4.1> UML-to-XML 변환기

그림 3.3에서 정의된 클래스에 대한 DTD에서 그림 4.2와 같은 XML 문서를 생성할 수 있다.

```
<Class>
  <Name> .. </Name>
  <IsActiv> true | false </IsActiv>
  <Attribute>
    <Name> .. </Name>
    <DataType> .. </DataType>
    <Visibility> "public" | "protected" | "private" </Visibility>
    <InitialValue> .. <InitialValue>
    <TargetScope> "instance" | "classifier" </TargetScope>
    <Changeable> "none" | "frozen" | "addonly" </Changeable>
    <Multiplicity> "1" | "*" | "0..1" | "m..n" </Multiplicity>
  </Attribute>
  <Operation>
    <Name> .. </Name>
    <Visibility> "public" | "protected" | "private" </Visibility>
    <ReturnType> ..</ReturnType>
    <Concurrency> "sequential" | "guarded" | "concurrent"  </Concurrency>
    <IsAbstract> true | false </IsAbstract>
    <IsLeaf> true | false </IsLeaf>
    <IsRool> true | false </IsRool>
    <OwnerScope> "instance" | "classifier" </OwnerScope>
    <Parameter>
      <Name> .. </Name>
      <DataType> . .</DataType>
      <DefaultValue> .. </DefaultValue>
      <Kind> "in" | "out" | "inout" | "return" </Kind>
    </Parameter>
  </Operation>
</Class>
```

<그림 4.2> 클래스에 대한 XML 문서

작성된 XML 문서에 새부속성을 추가적으로 입력 가능케 함으로서 보다 정밀한 표현이 가능하며, 모듈로서 사용한 XML DTD를 XML 스키마로의 확장이 가능하다.

5. 결론

본 논문에서는 UML 다이어그램의 저장 포맷으로 XML을 이용하였으며, UML 다이어그램의 입력으로 XML 문서를 출력하는 변환기를 제안하였다.
UML 다이어그램의 정보 저장이 자동화되고, XML의 특성을 살려 이 기종간의 정보교환의 이점을 살리고자 하였다. 이를 통해 UML의 확장성과 XML의 문서구조 활용의 극대화라는 측면의 이점을 가질 수 있다.
향후 연구 과제로는 XML DTD 뿐만 아니라 XML 스키마를 이용한 변환과, 다양한 다이어그램으로의 확장에 대한 연구가 이루어 져야 할 것이다.

참고문헌

[1] Grady Booch et al., "UML for XML Schema Mapping Specification", http://www.rational.com/media/uml/resources/media/uml_xmlschema33.pdf, 1999.
[2] 정경희, "차세대 웹 문서 표준 XML", 한국정보처리학회, Vol.6 No.3, P25-35, May, 1999.
[3] The source of the DOM specification and related specification, The W3C DOM Web page, http://www.w3.org/dom/, W3C, 1998.
[4] James Tauber, "Teach Yourself XML in 21 Days", SamsNet, 1998.
[5] The source of the XML specification and related specification, The W3C XML Web page, http://www.w3.org/xml/ ,W3C, 1998.
[6] Booch, Rumbaugh, Jacobson, The Unified Modeling Language User Guide, Addison-Wesley ,1997.
[7] 전세길, "UML 개발 과정의 멀티미디어 산출물 관리를 위한 XML 문서기반 객체 저장소의 설계 및 구현", 한국멀티미디어학회 추계학술발표논문집 (1999).