RDF 모델링 기법을 기반으로 한 MDL 모델링 기법

(A MDL Modeling Method based on RDF Modeling)

조 민 호 † 류 성 열 ††
(Min Ho Cho) (Sung Yul Rhew)

요 약 W3C에 의해 주도되고 있는 차세대 웹 기술은 HTML을 이용한 인간 중심의 웹 한계를 극복하는데 중점을 두고있다. 즉, 기존 웹상의 자료가 급격히 증가하고 있는 상황에서, 사람의 눈으로 내용을 확인해야 하는 HTML기반 기술은, 의미를 잃어가고 있으며, 이를 위한 새로운 대안이 필요하게 된 것이다. 이러한 이유로 사람의 눈 대신에, 기계가 정보를 찾아 주는, 기계 중심의 웹 분야가 새로운 대안으로 연구되고 있다.

본 논문에서는 기계 중심의 웹 환경 구현을 위하여 개발된 많은 방법 중에서 현재 가장 일반적으로 사용되고 있는 의미 서술 언어인 MDL을 소개하고, MDL 모델링 개념과 상세한 방법을 제시한다. 그리고 제안된 모델링 기법의 유용성을 검증하기 위하여 실무 사례를 기반으로 모델링 기법을 적용하였다. 마지막으로, 본 논문은 제안된 모델링 방법에 기반 한 MDL 에디터의 구현 화면을 제시함으로써, MDL 에디터 개발자에게 기술적 도움이 되고자 하였다.

키워드 : MDL, Semantic Web, 모델링, Editor

Abstract Web technology for next generation, led by W3C, is focusing on overcoming of limitation of Human-oriented Web. With circumstance of dramatic increasing information flow in current existing Web, human view based HTML technology is significantly loosing its value. To overcome limitation of Human-oriented Web, new alternative method is essentially required In this context, Machine-oriented Web which machine is searching for information, has been researched as a new alternative.

This thesis reviewed MDL (Meaning Definition Language) which is regarded as most generally using method for implementation of Machine-Oriented Web, and presented modeling concept and implementation method in detail. To verify the value of this presented implementation method, this thesis includes actual cases which has based on proposed implementation method .

As an end, we have tried to give a technical assistance to developer of MDL Editor by suggesting MDL Editor implementation display, which has been founded on the proposed modeling method.

Key words : MDL, Semantic Web, modeling, Editor

1. 서 론

본 논문은 W3C 표준에서 제시된 Resource Definition Framework(RDF)의 기본 개념을[1][2] 이용하여 Meaning Definition Language(MDL) 문서를[3] 모델링하는 방법을 제시함을 목적으로 한다.

현재의 웹 환경을 사람중심의 웹 환경이라고 할 때, 차세대 웹은 기계 중심의 웹 환경이라고 할 수 있다[4]. 즉, 기계가 사람을 대신하여, 웹의 정보를 찾고, 관련된 자료를 읽어서, 사용자에게 주거나 또는 연관된 처리를 수행할 수 있는 환경이 차세대 웹의 중요 기능이다[4]. 이러한 기능은 방대한 웹 정보의 활용에 있어서 필수적이라고 할 수 있으며, W3C에 의하여 Semantic 웹으로 분류되어 연구되고 있다. Semantic 웹은 XML을 기본으로 하고 있으며, RDF를 이용하여 다양한 자원을 정

† 비 회 원 : Openwave Korea Sales consultant
cho.min-ho@Openwave.com
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
syrhew@computing.soongsil.ac.kr
논문접수 : 2002년 4월 15일
심사완료 : 2002년 7월 11일

의하고, 데이타의 탐색을 위하여, 메타데이타를 이용하도록 구성되어 있다[4]. Semantic 웹의 실제 구현 모델이라고 할 수 있는 기계 중심의 웹은 RDF를 기반으로 하고 있으며, 정보 검색에 관련된 메타 데이타 기술은 RDF Schema/DAML+OIL/MDL 등 다양한 형태로 발전하고 있다[5].

기계 중심의 웹을 구축하는 많은 기술 중에서 XML 문서의 의미를 가장 정확하고 다양하게 표현하고, 활용할 수 있는 방법 중의 하나가 MDL이다. 하지만, MDL은 문법과 사용 방법은 간단한 반면, 제작된 문서를 읽기가 어렵고, 제작자에 따라 표현 형태가 매우 다양하게 나타날 수 있다는 단점이 있다. 더구나, 일반인들이 활용하기 위하여 필요한 각종 지원 도구의 개발이 미진하여, 기술 확산에 걸림돌로 작용하고 있다. 이러한 점에 착안하여, 본 논문에서는 Graham Klyne에 의해 제시된 RDF 모델링 방법을[6] 이론적 기반으로 하여, MDL을 모델링 하는 방법을 제안한다.

제안된 모델링 기법은 MDL의 실무적용을 위해 필요한 MDL Editor의 개발을 가능하게 하고, 제작되는 MDL문서의 형식을 정형화 함으로써, MDL기반의 기계 중심 웹 환경 확산에 기여할 수 있다.

2. 관련 연구

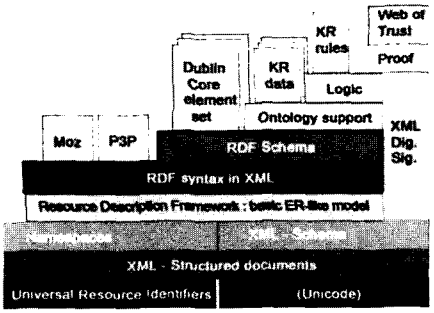
이번 장에서는 MDL 모델링에 대한 개념을 이해하기 위하여 Semantic 웹, XML, RDF의 관계를 설명하고, RDF 모델링의 개념과 방법을 제시한다. 마지막으로 MDL에 대한 기술적 설명을 제공한다

2.1 Semantic 웹, XML, RDF의 관계

W3C에서 제시하는 Semantic 웹에 관련된 자료에[7][8] 의하면, 향후의 웹은 현재의 환경과 같이 사람의 눈에 의해서 자료를 탐색하는 기능 외에, 특정 조건이 제시되는 경우에 해당되는 자료를 검색하고, 전달하는 모든 과정이 사람이 아닌 기계에 의해 대신 수행되는 환경이 추가되리라 생각된다[9]. 이러한 기능은 현재의 웹에 대한 활용도와 가치를 높이고, 보다 나은 정보 서비스를 가능하게 하는 기술이라고 할 수 있다.

이러한 환경의 성공적인 구현을 위해서는 관련된 기술들이 개발되어 제공되어야 하는데, 이를 위하여 eXtensible Markup Language(XML)을 기반으로 하는 기존의 기술(예: eXtensibel Style Language(XSL), Data Type Definition(DTD)..) 외에 Semantic 웹에서 사용하는 많은 자원(예: 메타데이타, User Query, ….)의 정의를 위한 RDF가 개발되었고, 실제적인 데이타 검색을 목적으로 메타데이타를 정의하기 위하여 RDF Schema, DAML+OIL, eXtensible Query Language (XQL), HornML 등 많은 기술들이 개발되었다.

앞에서 서술된 내용은 Eric Miller에 의한 Semantic 웹 *Symposium*자료에서 확인할 수 있으며, 그 중 대표적인 Semantic 웹 관련 기술 구성도를 그림 1에 제시하였다.



W3C Semantic Web activity, Eric Miller, International Semantic Web Workshop Symposium Jul30-august 1, 2001, http://www.w3.org/Talks/2001/07/30-sws/

그림 1 Semantic 웹 관련 기술 구성도

2.2 RDF 모델링의 개념

RDF 모델링은 본 논문의 기본적인 토대를 제시하는 개념이다[1][2]. 특히, MDL은 RDF를 기반으로 발전된 개념이기 때문에, MDL 모델링을 위해서는 RDF 모델링을 이해하는 것은 매우 중요한 일이다. RDF의 표준에 근거할 때, 표현해야 하는 정보의 최소 단위인 단위 Statement의 표기 방법은 아래와 같다[1].

[SubjectName] –propertyName→[ObjectName]

각 statement에 대한 이름을 재정의하기 위해서는 아래와 같은 방법을 사용한다.

Statement-id:[SubjectName] –propertyName→[ObjectName]

그리고, 각 Statement-id에 대한 속성을 지정하기 위해서는 아래와 같은 방법을 사용한다.

[Statement-id] –rdf:type→[rdf:Statement]
[Statement-id] –rdf:predicate→[propertyName]
[Statement-id] –rdf:subject→[SubjectName]
[Statement-id] –rdf:Object→[ObjectName]

2.3 Graham Klynn의 RDF 모델링 확장

앞 절에서 설명한 RDF 모델링 방법은 RDF를 이용하여 자원을 모델링하기 위한 방법이었다. 하지만, RDF는 매우 기본적인 수준의 정보를 표현할 수 있을 뿐, 보다 복잡한 형태의 정보를 표현하기에는 기능이 부족하다[3]. 이를 극복하기 위하여 제시된 Graham Klynn의

모델링방법[6]은 복잡한 내용(=Complex statement)은 궁극적으로 작은 낱알 수준(=Simple statement)으로 분할 될 수 있다는 McCarthy[10]와 Guha[11]의 논문에 이론적 바탕을 두고 있으며, 복잡한 내용의 서술에 대한 많은 아이디어를 제공한다.

Graham Klynn에 의한 RDF 모델링의 확장에 대한 이해를 위하여, 예를 제시하면, "내차는 Diesel형태의 연료를 사용한다"를 RDF의 형태로 보이면, [MyCar] -fuelType→[Diesel]과 같이 표시할 수 있다.

위의 상황을 조금 더 확대하면, "지금 내가 내차의 엔진에서 쓰이는 연료의 종류에 대한 것을 서술하기를 원한다"의 예로 확장할 수 있고, 이러한 것을 RDF의 형태로 보이면, *[MyCar] -hasEngine→[MyCarEngine] -fuelType→[Diesel]*과 같이 표시할 수 있다.

앞의 예에서 확인할 수 있는 바와 같이, RDF에서 제공하는 기본적인 표기법은, 실제 복잡한 상황에서는 표현하기가 적당하지 않기 때문에,[1] 모듈화와 트리의 개념을 적용하여 표기하는 방법을 바꾼 것이 Graham Klynn에 의한 RDF 모델링의 확장이다. Graham Klynn의 확장된 모델에서는, 최종적인 Object는 가독성을 높이기 위하여 []대신에, ""으로 표기한다.[2]

보다 나은 이해를 위하여, 앞에서 언급한 예인 "지금 내가 내차의 엔진에서 쓰이는 연료의 종류에 대한 것을 서술하기를 원한다"의 내용을 Graham Klynn의 확장 모델링으로 표시하면 아래와 같다.

```
[MyCar] -description→
{
    [Engine] -description→
      {
        [FuelUsed] -fuelType→"Diesel"
    }
}
```

위의 표기법은 이전에 RDF의 표시법에 근거하여 제시한 것보다, 체계적이며, 보기에 편하고, 나중에 필요한 사항이 발생하는 경우에도 손쉽게 수정할 수 있다. 예로써, 위와 같은 표기 후에, 나의 차에 대한 Door의 특성을 추가하고 싶은 경우에는, Door가 차에 속하는 속성이기 때문에 다음과 같이 표시할 수 있다

```
[MyCar] -description→
{
    [Engine] -description→
    {
        [FuelUsed] -fuelType("Diesel"
    }
    [Door] -description(
      {
      }
}
```

또, 위와 같은 표시 후에, 내 차의 종류를 서술하고 싶은 경우에는 다음과 같이 표시할 수 있다.

```
[MyCar] -isa→[FordEscort]
[      ] -description→
{
    [Engine] -description→
    {
      [FuelUsed] -fuelType→"Diesel"
    }
    [Door] -description→
    {
    }
}
[House] -description→
{
}
```

위와 같은 서술 방식을 사용하면, 기존의 RDF에서 제시하는 방법이 Complex Statement를 서술할 때 가지는 단점을 해결할 수 있다.[3]

이 절에서 설명한 Graham Klynn의 방법은 본 논문의 주제인 MDL 모델링을 위한 기본적인 아이디어[4]를 제공하고 있으며, 본 논문의 논리적 기반을 제공하고 있다.[5]

2.4 Charteris사의 MDL Draft 2.02

MDL[3]은 XML로 되어 있는 문서가 의미하는 바를 정의하고, 문서에서 그러한 의미가 어떻게 표현되어 있는지를 정의하는 것을 목적으로 만들어진 언어이다

Semantic 웹의 핵심 구성요소는 RDF와 RDF Schema이다.[6] Semantic 웹에서는 RDF와 RDF Schema를 이용하여 다양한 자원을 정의한다. 활용적 차원에서 보면, RDF와 RDF Schema는 XML문서가 의미하는 바를 정의할 수는 있지만, 어떻게 그런 의미를 나타내는 가를 정의하지는 못한다

이러한 점을 극복하기 위하여 MDL은 Xpath를 사용하여 XML문서가 전달하는 의미를 명확히 정의한다. 즉, XML문서가 전달하는 의미를 정의하기 위해서 각

1) 다르게 말하면, 작성하기도 어렵고, 작성된 결과물을 읽기도 어렵다는 의미

2) 위의 예에서 [Diesel을 "Diesel"로 표시한다.

3) 이절의 앞부분을 참조

4) Tree구조와 {}기반의 Module화 그리고 중복서술 방식을 말함

5) 구체적으로 Graham Klynn의 Modeling방법이 어떻게 적용되었는지에 대한 것은 3.4를 참조한다.

6) 그림 1 참조

노드의 정보를 정의하는 것 뿐만 아니라, 사용자의 의도에 맞는 값을 여러 엘리먼트 속에서[7] 올바르게 뽑아내기 위해서 XPath에 근거한 탐색을 추가하였다. 다시 정리하면, 같은 엘리먼트라고 하여도, path가 다르면, 다른 의미를 가진다는 가정 하에서 탐색하는 과정이 필요하다. 이러한 점이 MDL이 가지는 핵심 요소이다.

3. 구조적 MDL 모델링

이번 장에서는 본 논문에서 제시하고자 하는 MDL의 구조적 모델링 기법에 대하여 상세히 서술한다. 그리고, 제시된 모델링 기법을 다양한 형태의 XML 문서에 대응하여, 제시된 모델링 기법의 유용성을 보이고(4장), 최종적으로 제시된 모델링 기법을 활용하여, MDL Editor의 제작 사례를 소개한다(5장).

3.1 MDL 모델링의 개념

MDL은 기본적으로, Semantic 웹 환경의 구현을 위하여 개발된 기술이고, 이러한 기술을 배경으로 *Semantic* B2B, 자연어 기반의 Query 등과 같은 다양한 분야에서 활용되고 있는 기술이다. 하지만, MDL의 실질적인 구현을 위해서는 웹의 각종 정보마다(예: *XML, XHTML, HTML* 문서) 정보의 의미를 서술하는 MDL을 제작해야 하는데, MDL작성을 위한 표준적인 방법이 없고, 실제 업무에서 손쉽게 사용할 수 있는 *MDL Editor*의 개발이 미진하여, 확산에 걸림돌로 작용하고 있다. 이러한 이유로, MDL Editor를 개발하기 위한 전 단계로, MDL 문서를 모델링하는 방법을 제공하고자 하는 것이 본 논문의 핵심이다.

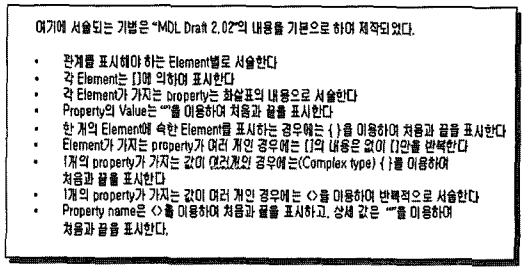
본 논문에서 제시하는 MDL 모델링 방법은 Graham Klyne에 의해 연구된 기존의 RDF 모델링 방법에서 사용된 기법들을, MDL 모델링에 적용하고자 하는 것이다.[8]

3.2 MDL 모델링 방법

본 논문에서는 *MDL* 모델링에 대한 설명을 위해, 먼저, 모델링을 위한 기본적인 가이드라인을 제시하고, 그 다음으로, 실제 사용을 위한 템플릿을 제시하였다. 마지막으로, 주어진 가이드라인과 템플릿의 사용법에 대한 설명을 위해, 실제 예를 가지고 모델링을 수행하는 상세한 과정을 제시하였다.

참고로 본 논문에서 제시하는 *MDL* 모델링 가이드라인은 Charteris사에서 발행한 MDL Draft 2.02를 기반으로 제작하였다[3].

먼저, 실제적인 모델링을 수행하기 위하여 필요한 모델링 가이드라인의 상세한 내용은 이해를 위하여 그림 2에 요약하였다.


여기에 서술되는 기법은 "MDL Draft 2.02"의 내용을 기본으로 하여 제작되었다.

- 관계를 표시해야 하는 Element별로 서술한다
- 각 Element는 []에 의하여 표시한다
- 각 Element가 가지는 property는 화살표의 내용으로 서술한다
- Property의 Value는 ""를 이용하여 처음과 끝을 표시한다
- 한 개의 Element에 속한 Element를 표시하는 경우에는 { }을 이용하여 처음과 끝을 표시한다
- Element가 가지는 property가 여러 개인 경우에는 []의 내용은 같이 []안을 반복한다
- 1개의 property가 가지는 값이 Data인 경우에는(Complex type) { }를 이용하여 처음과 끝을 표시한다
- 1개의 property가 가지는 값이 여러 개인 경우에는 <>를 이용하여 반복적으로 서술한다
- Property name은 <>를 이용하여 처음과 끝을 표시하고, 상세 값은 ""를 이용하여 처음과 끝을 표시한다.

그림 2 MDL 모델링을 위한 가이드라인

그림 2에서 제시하는 가이드라인은 각 XML문서의 엘리먼트 또는 어트리뷰트가 의미하는 바를 MDL로 표현할 때, 지켜야 하는 기본 지침이다. 위의 지침을 근거로 MDL을 서술하면, MDL 문서를 트리 구조의 형태로 표현할 수 있으며, 이러한 형태는 GUI를 기반으로 하는 *MDL Editor*의 개발을 용이하게 한다.

그리고, 모델링 가이드라인 외에도, 실제적인 모델링을 수행하고자 하는 사용자의 이해를 위하여 별도의 템플릿을 준비하였으며, 실제적인 내용은 그림 3을 참고할 수 있다.

그림 3에서 제시하는 템플릿은 그림 2에서 제시하는 가이드라인의 실제 적용 예를 보여주고 있다. 그림 3의 각 부분은 XML문서의 의미를 표현하고 있으며, 의미의 표현을 위하여 연관된 엘리먼트와 어트리뷰트는 무엇이며, 연관 관계는 무엇인가를 가이드라인에서 제시하는 방법에 따라 서술하게 된다.

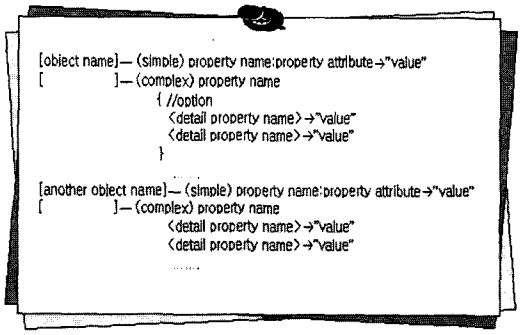


그림 3 MDL 모델링의 기본적인 형태

7) 엘리먼트의 이름이 같지만, 위치에 따라 다른 의미를 표현할 수도 있다.

8) 3.4의 내용을 참조

3.3 MDL 모델링 상세 과정

이번 장에서는 3.2에서 제시한 방법의 이해를 높이고, 실제적으로 MDL 모델링을 하고자 하는 실무자를 위하여 3.2에서 제시한 방법을 기반으로 하여, XML문서의 의미를 MDL로 모델링하는 단계를 상세히 서술한다. 이를 위하여 간단한 예를 들어보면

학교를 서술하는 XML문서가 있고, XML문서가 학생을 의미하는 pupil 엘리먼트를 가지는 경우에, XML문서는 아래와 같은 형태를 가지게 된다.

```
<school name="Soongsil Univ">
        <pupil name="minho"/>
        <pupil name="hyunhoon"/>
</school>
```

위의 예에서 제시한 XML문서에서 학생을 의미하는 pupil 엘리먼트는 School 엘리먼트와 "학교에 다니는 학생(attends)"의 관계를 가지고 있다면, 이러한 관계는 MDL에서 다음과 같이 표시될 수 있다

```
< Element context='/school/pupil'>
    <me:object class='student'>
        <me:inclusion>9)
            <me:condition assoc="attends"
    obj1="student" obj2="school"/>
        <me:inclusion>
    </me:object>
</ Element>
```

이러한 MDL문서를 MDL 모델링을 이용하여 모듈화와 트리의 개념을 적용하여 모델링하면 아래와 같다.

```
[pupil] -path→"/school/pupil"
[      ] -object→"student"
[      ] -condition:inclusion→
            {
                <assoc>-→"attends"
                <obj1>-→"student"
                <obj2>-→"school"
            }
```

이러한 모델링기법은 3.2에서 설명한 MDL 모델링기법의 개념을 실제 적용한 것으로써, 위의 MDL 모델링이 나오기 까지의 단계를 요약하면 그림 4와 같다.

보다 자세히 설명하면,

첫째, "서술하고자 하는 Element를 선정한다"에서, 서술하고자 하는 내용이 pupil 엘리먼트가 School 엘리먼트에 다닌다는 의미이므로, Pupil서정

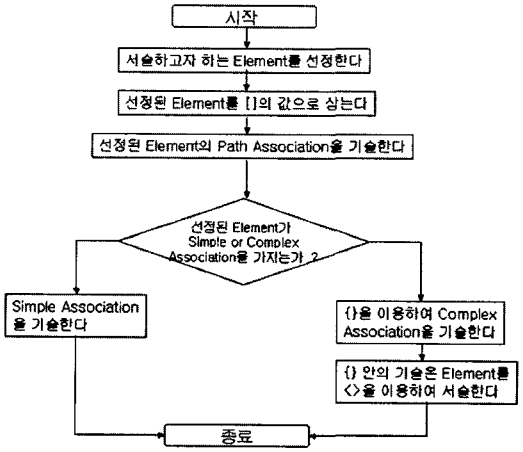


그림 4 MDL 모델링 단계 요약

둘째, "선정된 Element를 []의 값으로 삼는다"이므로 [pupil]과 같이 표현할 수 있다

셋째, "선정된 Element의 Path Association을 기술한다"이므로, [pupil] -path→"/school/pupil"와 같이 표시할 수 있다.

넷째, 표준 Modeling단계에는 없지만, pupil을 향후, 질의에서 어떠한 이름으로 사용할 지를 정해야 하는 단계가 필요하다면, [ ] -object→"student"처럼 명시한다.

다섯째, 관계의 표시는 Complex이므로 {}을 이용하여 싸고, {}에 의해 싸여진 요소들은 <>을 이용하여 묶는다.

3.4 MDL 모델링 방법과 RDF 모델링 방법의 비교

기본적으로 MDL 모델링은 RDF 모델링과 Graham에 의해 확장된 RDF 모델링의 개념을 확대하여 개발되었다. 그런 의미에서 그림 2와 그림 3에서 제시된 방법이 RDF 모델링과 어떠한 연관관계가 있는가를 상세히 설명한다.

연관관계를 논하기 전에 먼저 RDF와 MDL의 용어의 차이를 정리할 필요가 있다. 먼저, RDF는 자원, Property, Object의 3가지 구성요소로 이루어 지지만, MDL은 엘리먼트/어트리뷰트, Property, Value의 3가지 구성요소로 구성된다. 이들 각자의 이름은 다르지만, 모델링이라는 개념에서 볼 때,[10] 자원은 엘리먼트/어트리뷰트와, Property는 Property와 Object는 Value와 연결하여 표

9) inclusion은 유일함을 의미하는 지시어이다. 전체적인 면에서 무시할 수 있음.

10) 실제적인 모델링에 대응되는 구성요소가 같고, 실제적인 의미도 동일하다.

시할 수 있다는 점이 MDL 모델링의 기본 개념이다.

이러한 기본 개념을 전제로 하여, 그림 2와 그림 3에서 제시한 MDL 모델링은 RDF 모델링으로부터 아래의 개념을 도입하여 사용한다.

첫째, RDF 모델링에서 사용하였던, Resource, Property, Object의 표기법을 그대로 이용한다. 실제 예로써, Resource는 엘리먼트/어트리뷰트를 나타내므로 []으로 묶고, Property는 동일하므로 화살표의 값으로 표시하며, Object는 Value를 나타내므로 " "를 이용하여 표시한다.

둘째, Rosource가 여러 특성을 가지는 경우에, 그림 3에서 보여주는 바와 같이 []을 반복하되, []안의 값은 중복해서 명시하지 않는다. 이러한 경우는 MDL에서 동일 엘리먼트/어트리뷰트가 여러 개의 특성을 가지는 경우와 같다고 볼 수 있으며, MDL 모델링에서도 []안의 값을 중복해서 명시하지 않는다.

셋째, 하나의 property에 대해 여러 개의 서술이 존재하는 경우에는 {}을 이용하여 묶어준다. 이러한 경우는 하나의 property에 대하여 MDL에서 여러 개의 Value가 존재하거나, 또는 여러 Value를 가지는 부분과 연계되는 경우와 같다고 볼 수 있다. 실제 MDL에서도 여러 개의 값이 존재하는 경우에 {}을 이용하여 묶는다

위와 같은 세 가지의 특성은 RDF 모델링의 기본을 이루는 것으로써, MDL 모델링에서도 그대로 적용된다. 그리고, RDF에는 해당되지 않지만, RDF 모델링 방법을 MDL에 적용할 때 발생하는 예외적인 경우에는 MDL 모델링을 위하여 별도의 가이드라인을 추가하였다

보다 자세히 설명하면, 하나의 Property에 대응되는 Value가 한 개가 아닌 경우에는 {}을 이용하여 묶는 다는 점은 RDF에서도 언급된 바 있다. 하지만, 여러 개의 Value가 또다른 구조와 연관관계를 가지는 경우(예:하나가 다른 하나와 특정 관계를 가지는 경우 …)에는 이들을 표현하는 방법이 없다. 이러한 경우에는 RDF에는 없지만, <>을 이용하여 엘리먼트/어트리뷰트를 표시하고, 화살표를 이용하여 Property를 나타내며, Value는 ""을 이용한다. 그리고 이러한 값들은(<>으로 표시되는 것들) RDF와 마찬가지로 {}을 이용하여 묶는다

4. MDL 모델링 사례

3장에서 제시한 MDL 모델링에 대한 기법은 실무적인 환경에서도 적용 가능한 기법이다. 이를 입증하기 위하여 3장에서 제시한 MDL 모델링을 실제 업무에 적용하여 보고자 한다. 사례를 위하여 준비된 XML문서는 "*Professional XML MetaData*"[11]에서 발췌하였으며, 실제로 MDL이 적용되어 운영되고 있는 사례의 일부이다[13].

먼저 사례로 제시된 XML문서를 보면 그림 5와 같다. 제시된 XML문서는 학교에서 강의하는 과목과 교수에 대한 정보를 가지고 있다.

```
<?xml version="1.0" encoding="utf-8"?
<college><faculty>
<lect lectID="1024" fName="John" sName="Budden">
<title>reader</title>
<dob>1948</dob>
<appt>1995</appt>
<tenure>yes</tenure>
<crs crsId="c101" name="Physics 1" credits="120pMark
        = "60" year="1"/>
<crs crsId="c104" name="Physics 2" credits="150pMark
        = "50" year="2"/>
<crs crsId="c171" name="Applied Mathscredits="100" pMark
        = "70" year="3"/>
<crs crsId="c180" name="Analysis" credits="60pMark
        = "60" year="2"/></lect>
…. 동일한 내용이 계속 반복되므로, 중간 생략 ….
```

그림 5 실무 모델링 사례를 위한 XML문서

그림 5의 XML문서에 대한 이해를 높이기 위하여 그림 5의 XML문서를 구성하는 중요 엘리먼트의 의미를 요약하면 다음과 같다.

```
<lect lectID="1024" fName="John" sName=
            "Budden"> // 교수에 대한 정보
<title>reader</title>
<dob>1948</dob>
<appt>1995</appt>
<tenure>yes</tenure>          } //과정에 대한 추가 속성
<crs crsId="c101" name="Physics 1" credits
        ="120" pMark="60" year="1"/>
    //교육과정에 대한 정보
```

그림 5에서 제시한 XML문서의 의미를 MDL 모델링 기법을 이용하여 모델링하면 다음과 같다. 실제로 그림 4에 제시된 절차를 따라서, 그림 5의 XML문서를 직접 모델링할 수 있으며, 이러한 모델링 기법은 실무에 적용하는 경우에 매우 유용하다.

```
lect] -path→"/college/faculty/lect"
        // 엘리먼트와 Property, Value 명시
[   ] -object→"teacher"
        // 엘리먼트의 추가적인 Property 명시 ( [] )
    {   // 엘리먼트의 Property가 여러 개의 값을 가지
        는 경우 ({})
      <fName> -path("/college/faculty/lect/@fName"
        // 추가속성 명시(<>)
```

11) Wrox published, 2001

```
          <       >-object("firstName"
          <sName>-path("/college/faculty/lect/@sName
          <       >-object("surname"
        }
[title]-path→"/college/faculty/lect/title"
[    ]-object→"title"
[    ]-property→"teacher"
[dob]-path→"/college/faculty/lect/dob"
[    ]-object→"yearOfBirth"
[    ]-property→"teacher"
[appt]-path→"/college/faculty/lect/appt"
[    ]-object→"yearOfAppointment"
[    ]-property→"teacher"
[tenure]-path→"/college/faculty/lect/tenure"
[    ]-object→"tenure"
[    ]-property→"teacher"
[crs]-path→"/college/faculty/lect/crs"
[    ]-object→"course"
[    ]-association→"teaches"
    {
      <teacher>--objecttoAssociation→"crs"
      <       >--associationToObject→"parent::lect"
      <course>--abjectToAssociation→"."
      <       >--associationToObject→"."
      {   // 추가적인 속성이 또 다른 여러 속성을 가지는 경우
        <name>-path→"/college/faculty/lect/crs/@name"
        <       >-object→"name"
        <credits>-path→"/college/faculty/lect/crs/@credits"
        <       >-object→"credits"
        <pMark>-path→"/college/faculty/lect/crs/@pMark"
        <       >-object→"passMark"
        <name>-path→"/college/faculty/lect/crs/@year"
        <       >-object→"year"
      }
    }
```

위와 같이 모델링된 XML문서의 의미는, 다른 말로 하면, XML문서를 구성하는 엘리먼트간의 관계는 MDL로 직접 변환될 수 있으며, 실제적으로 변환된 MDL문서의 일부를 제시하면 그림 6과 같다

이번 장에서 보여준 바와 같이, 실제업무에서 제작된 XML문서를 MDL을 기반으로 하는 Semantic 웹 환경에서 효율적으로 이용하고자 하면(예: 자연어 질의를 통한 자료 검색, 질의에 대응하는 프로그램의 수행 등), 실제적으로 XML문서를 구성하는 각 엘리먼트와 어트리뷰트간의 의미 관계를 MDL로 표현하는 과정이 필요하다. 그리고, 실제 환경에서 작성되는 MDL은 제작자의 의도에 따라 다양한 형태로 표현되며, 이러한 것은 MDL의 활용을 위해 MDL을 Access하는 프로그램을 제작할 때, 프로그램이 제작자의 표현 형태를 반영해야 한다.

이러한 점에서, 본 논문에서 제시한 방법과 같이, XML문서가 주어지면, 이를 일정한 규정에 따라 모델링하고, 모델링이 끝나게 되면, MDL문서는 자동적으로 생성하게 하는 방법은 MDL의 표현 방식을 통일하고,실무적 차원에서 MDL을 이용하는 프로그램의 제작시에도 많은 노력의 경감 효과를 볼 수 있다.

결론적으로, 이번 논문에서 제시하는 모델링방법은 MDL의 일반화를 위한 MDL Editor의 제작에도 많은 기여를 하리라 예상되지만, 또다른 점에서, MDL의 표현 방법을 통일하는 효과를 가지게 하여, 실무 프로그램의 제작 효율을 높이는데 기여하게 될 것이다.

```
.. 윗부분 생략
    <Elemen context="/college/faculty/lect">
        <me:object class="teacher"/>
    </Elemen>
    <Attribut context="/college/faculty/lect/@fName">
        <me:property class="teacher" property="firstName">
            <me:find objectToProperty="@fName"/>
        </me:property>
    <Attribut>
    <Attribut context="/college/faculty/lect/@sName">
        <me:property class="teacher" property="surName">
            <me:find objectToProperty="@sName"/>
        </me:property>
    <Attribut>
    <Elemen context="/college/faculty/lect/title">
        <me:property class="teacher" property = "title">
            <me:find objectToProperty="title"/>
        </me:property
    </Elemen>
... 아랫부분 생략
```

그림 6 MDL 모델링 결과에 근거한 MDL Document

5. MDL 모델링의 활용

4장에서 제시된 MDL 모델링은 MDL을 적용하는 실무에서 다양하게 사용될 수 있다. 하지만, 활용의 측면에서 가장 유용하게 사용될 수 있는 곳 중의 하나가, MDL Editor이다. 이런 점에서 4장에서 제시된 MDL 모델링 기법을 이용하는 MDL Editor에 대하여 논하고자 한다

5.1 MDL Editor

MDL Editor는 MDL Document를 자동적으로 생성하여 주는 도구이다. XML을 활용하기 위해서는 XML Editor가 필요한 것처럼, MDL Editor는 MDL을 활용하기 위한 필수적인 도구라고 할 수 있다. MDL Editor는 MDL의 작성을 쉽게 하고, 작성된 MDL의 수정을 지원하며, MDL문서의 전체 구조에 대한 종합적인 관리 기능을 지원하는 GUI기반의 도구를 말한다.

이러한 MDL Editor는 MDL의 일반화와 표현의 다양화를 정형화 시켜주는 효과를 가져서, MDL의 실무 적용 및 확산을 빠르게 하기 위한 필수적인 도구라고

할 수 있다

5.2 MDL Editor의 활용 방법

Semantic 웹을 구현하고자 하는 경우에, 구현 방법으로 MDL을 선택한 경우라면, Contents를 서술하는 모든 자료(예: HTML, XML 자료…)에 대한 MDL을 작성하여야 한다. 하지만, MDL자체를 작성하는 것이 Contents를 구성하는 자료의 성격에 따라서 상이하고, 동일한 의미를 서술한다고 하여도, MDL 문서의 형태가 작성자의 취향에 따라서 달라지게 된다. 이런 점을 고려하여 볼 때, MDL의 작성을 쉽게 하고, 작성된 MDL의 가독성을 높이기 위해서는, MDL Editor가 필요하다. 또한, 기존에 작성된 MDL의 수정 보완과 관리에 있어서도 MDL Editor는 매우 중요한 역할을 하게된다. 이러한 업무절차는 그림 7에 요약되어 있다

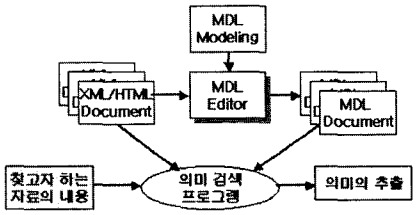


그림 7 MDL Editor의 역할

그림 7에서 제시한 바와 같이 MDL Editor는 MDL 기반의 Semantic 웹환경 구현의 가장 중요한 역할을 수행하고 있다. 즉, XML과 MDL간의 중간 역할을 수행하여, 전체적인 Contents의 의미 정의를 표준화하는데 기여하는 도구라고 할 수 있다.

5.3 MDL Editor의 개발 사례

본 논문에서 제시된 모델링 방법의 평가를 위하여, MDL 모델링 자체를 이용하는 대표적인 예라고 할 수 있는 MDL Editor의 개발 사례를 그림 8에 제시하였다. 그림 8에 제시된 MDL Editor는 제품으로 완성된 것은 아니지만, 본 논문에서 제시된 MDL 모델링을 활용하여 제작될 MDL Editor의 모습을 확인하고, 제시된 모델링 방법의 의미와 응용 범위를 확인할 수 있을 것이다.

그림 8에서 제공한 화면은 XML문서가 작성되어 있다는 전제 하에서, XML문서의 의미를 서술하기 위한 목적으로 MDL Editor를 사용하는 예이다.

사용자는 본 논문에서 제시한 모델링기법을 적용하여, MDL Editor의 왼쪽 상단 윈도우에 Tree구조의 형태로 의미를 서술하고, 서술이 끝나면, *MDL Tree/Generation* Code메뉴를 선택하여 자동적으로 MDL문서를 생성할 수

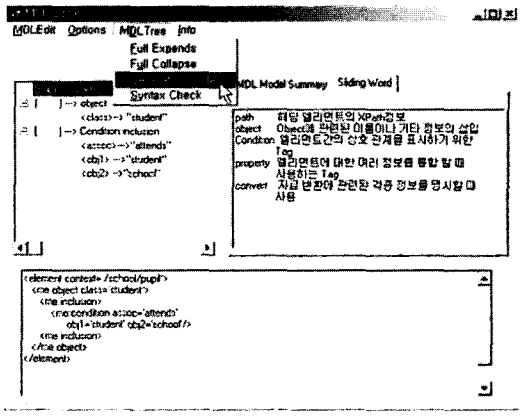


그림 8 MDL Editor의 화면 예

있다. 또한 기존의 MDL문서를 읽어서, Tree구조의 형태로 보면서, 수정, 삭제, 보완할 수 있는 기능을 제공한다. 향후에 최종적인 제품의 완성 단계에서는 각각의 XML과 MDL문서를 통합 관리하는 기능이 추가될 것이며, 하나의 XML에 대하여, 의미에 따라 여러 개의 MDL을 정의하는 기능도 추가될 예정이다.

5.4 MDL Editor의 개발을 기반으로 한 MDL Modeling 기법의 평가

이번에는 제안된 MDL Editor의 개발 경험을 기준으로 하여 볼 때, 본 논문에서 제안한 MDL Modeling 방법의 유용성을 정리한다.

(1) XML문서가 의미하는 것을 Tree구조를 기반으로 표현함으로써 작성된 MDL의 가독성을 높일 수 있었다. 이러한 점은 실제로 다른 사람에 의해 만들어진 MDL 문서를 유지/보수해야 하는 경우에 매우 유용하였다.

(2) Tree구조로 표현된 것을 자동적으로 MDL문서로 생성함에 의하여, 작성자의 취향에 따라서, 동일한 MDL문서가 다르게 표현되는 것을 방지할 수 있었다.

(3) 한번 제작된 MDL문서에 다른 의미를 추가/삭제하고자 하는 경우에, Tree 구조를 추적하여 필요한 부분만을 변경하고, MDL Code를 새로 생성함으로써, 관리의 노력이 줄어들었다.

6. 평 가

본 논문에서 제시된 MDL 모델링 방법은 MDL자체가 RDF를 기반으로 개발된 기술이라는 점에서 Graham Klynn의 RDF 모델링 논문에서 제시한 Tree기반의 모델링 기법을 이론적 기본으로 하였다. MDL 자체가 최근에 개발된 기술이고, 모델링에 대한 관련된 연구가 없어서,

본 논문에서 제시한 방법에 대한 것을 다른 방법과 비교할 수 는 없다. 하지만, XML 문서가 가지는 의미를 MDL의 관점에서 모델링하기 위한 기본적인 원칙과 Template 그리고 작성하는 과정의 Flow-chart, 상세한 작성 단계의 설명, 마지막으로 실제 적용된 XML문서를 대상으로 한 모델링 기법의 적용을 제시하였기 때문에, 실무에서 MDL을 활용하고자 하는 실무자 또는 MDL Editor 개발자들에게 MDL문서를 정형화하는 것에 관련하여 매우 가치 있는 자료가 되리라 생각한다. 그리고, 제안된 모델링 방법을 기반으로 실무에서 활용할 수 있는 MDL Editor의 기본적인 형태를 제시함으로써, 본 논문에서 제시한 모델링 방법의 유용성을 입증하였다.

7. 맺는말

본 논문은 MDL이 Semantic 웹 환경을 구현하기 위한 좋은 기술이라는 개념에서 출발하여, 실무적 차원에서 MDL의 확산을 위해서는 별도의 도구가 필요하다는 점을 인식하고, 이를 위한 기본적인 방법으로써, 모델링 방법을 제안하고자 한 것이었다. 모델링 방법의 개발을 위해 RDF 모델링 방법을 제안한 Graham Klynn의 논문을 기본으로 하였으며, 필요한 부분은 일부 보완하였다. 제안된 모델링 방법의 기술적 유용성을 검증하기 위하여 실제 프로젝트에서 사용되었던 XML문서를 대상으로 모델링 방법을 적용하고, 제안된 Modeling 기법을 활용한 MDL Editor의 개발 사례도 제시하였다.

본 논문에서 제시된 모델링 기법은 MDL Draft 2.02를 기본으로 제작되었으나. 모델링 기법 자체가 모듈화와 트리 구조의 개념이 적용되어 있기 때문에, 향후, MDL에 기능이 추가되는 경우에도, 큰 무리 없이 확장 적용될 수 있다.

MDL은 Charteris사에 의해 실무에 적용되고 있고, 상세한 사양이 웹에서 화이트 페이퍼의 형태로 배포되고 있으며, 이미 30여개의 회사에 적용되어 활용되고 있는 기술이다[12]. 마지막으로, 본 논문의 가치는 MDL의 활성화를 위해 필요한 MDL Editor제작의 기반을 다진 것과, 이를 바탕으로 MDL이 실무적 차원에서 많이 활용 될 수 있는 이론적 기반을 제공한 데서 찾을 수 있을 것이다

참 고 문 헌

[1] W3C, "RDF Model and Syntax Specification," at URL: www.w3c.org, 1999.
[2] W3C, "RDF Schema Specification 1.0," at URL: www.w3c.org, 2000.
[3] Robert Worden, "A Meaning Definition Language Draft 2.02," at URL: www.Charteris.com, 2001.
[4] Tim Berners-lee, James Hendler, Ora Lassila, "The Semantic Web", at URL: www.scientificamerican.com/2001/0501issue/0501berners-lee.html., 2001.
[5] Harold Boley, Stefan Decker,Micahel Sintek, "Tutorial on Knowledge Markup Techniques," ECAP 2000 Berlin, 22. Aug, 2000.
[6] Graham Klynn, "Information Modeling using RDF," at URL: public.research.mimesweeper.com/RDFInfomodeling.pdf, 2001.
[7] Stefan Decker, Sergey Melnik, "The Semantic Web:The Roles of XML and RDF", IEEE Internet Computing, September-October 2000.
[8] Tim burners-Lee, "Semantic Web Road Map", at URL www.w3c.org/DesignIssues/Semantic.html, 2001
[9] Jon Bosak, Tim Bray, "XML and the Second-Generation Web", at URL: www.scientificamerican.com/1999/0599issue/0599bosak.html, 1999.
[10] John McCarthy, "Notes on Formalizing Context, Computer Science Department, Standford University, at URL: www-formal.Stanford.edu. 1990.
[11] Ramanathan V. Guha, "Contexts: A Formalization and Some Applications," Stanford PhD Thesis, at URL: www-formal.Stanford.edu,1991.
[12] Charteris HomePage, at URL www.charteris.com, 2001.
[13] Kal Ahmed, other 8 person, "Professional XML Meta Data," Wrox, 2001.

조 민 호
1963년생. 1989년 인하대 산업공학과 졸업. 1998년 숭실대 정보과학 대학원 졸업. 현재 숭실대 박사과정 재학중. 1989년 ~ 2000년 HP Korea Sales consultant. 2000년 ~ 현재 Openwave Korea Sales consultant. 관심분야는 XML, Semantic Web

류 성 열
1997년 아주대학교 컴퓨터학부(공학박사). 1997년 ~ 1998년 George Mason Univ. 교환교수. 1981년 현재 숭실대학교 컴퓨터학부 교수. 1998년 현재 숭실대학교 정보과학대학원 원장. 1998년 ~ 현재 숭실대학교 전자계산원 원장. 관심분야는 리엔지니어링, 분산 객체 컴퓨팅, 소프트웨어 재사용