시맨틱웹 기술과 활용방안

The SemanticWeb Technology and its Applications

오 삼 균(Sam-Gyun Oh)*

초 록

시맨틱웹은 기계가독형 정의에 기반한 정보의 연계를 통해 웹 자원을 지식화함으로써 정보의 효율적 검색, 통합, 재사용을 도모하는 새로운 기술이다. 시맨틱웹의 구축은 자원에 불변 고유식별자를 부과하는 URI 체제, 각 정보기관에서 생성되는 요소와 속성의 의미 충돌을 방지하는 XML 네임스페이스, 메타데이터 스키마를 활용한 호환적 자원기술을 가능하게 하는 RDF, 메타데이터 요소 및 이와 연관된 클래스와 속성 관계 정의의 기반이 되는 RDF 스키마, 그리고 RDF 스키마 위에 논리적 추론과 표현력을 강화한 웹 온톨로지 언어 DAML+OIL 및 그 건조자(constructors)를 삭제 또는 수정 보완한 OWL (Web Ontology Language) 등의 여러 핵심 개념과 기술을 필요로 하는 작업이다. 이 논문은 이러한 개념과 기술의 점진적 발전 양상을 개괄 설명하고, XML/RDF 스키마를 기반으로 메타데이터 요소들을 정의할 경우 도출할 수 있는 상호운용성과 온톨로지의 다양한 활용 방안 등을 고찰한다.

ABSTRACT

The Semantic Web is a new technology that attemps to achieve effective retrieval, automation, integration, and reuse of web resources by constructing knowledge bases that are composed of machine-readable definitions and associations of resources that express the relationships among them. To have this kind of Semantic Web in place, it is necessary to have the following infrastructures: capability to assign unchangeable and unique identifier (URI) to each resource, adoption of XML namespace concept to prevent collision of element and attribute names defined by various institutions, widespread use of RDF to describe resources so that diverse metadata can be interoperable, use of RDF schema to define the meaning of metadata elements and the relationships among them, adoption of DAML+OIL that is built upon RDF(S) to increase reasoning capability and expressive power, and finally adoption of OWL that is built upon DAML+OIL by removing unnecessary constructors and adding new ones based on experience of using DAML+OIL. The purpose of this study is to describe the central concepts and technologies related to the Semantic Web and to discuss the benefits of metadata interoperability based on XML/RDF schemas and the potential applications of diverse ontologies.

키워드: 시맨틱웹, 네임스페이스, 메타데이터 상호운용성, 온톨로지, Semantic Web, Namespaces, Metadata Interoperability, Ontology

---

* 성균관대학교 문헌정보학과 부교수(samoh@skku.ac.kr)

■ 논문 접수일 : 2002. 11. 29
■ 게재 확정일 : 2002. 12. 10

1 연구의 배경 및 목적

웹의 실용화에 견인적 역할을 한 대표적 인물 버너스리는 웹이 제안된 본래의 목적은 사람 사이의 원활한 커뮤니케이션과 더불어 컴퓨터 시스템들 간의 효과적 커뮤니케이션이라고 지적하였다 (Bernes-Lee 2000). 이 목적의 실현은 인터넷 자원을 소화할 수 있는 하부구조의 정립을 전제한 매우 복잡하고도 시급한 과제라 할 것이다.

단일 인터페이스로 모든 유형의 정보(텍스트, 음악, 동영상, 이미지)에 접근하는 방법과 용이성과 융통성을 그 특징으로 하는 HTML의 광범위한 수용으로 인터넷 정보량은 폭발적으로 증가했고 그 결과 전문색인 기반 검색엔진의 효율은 현격히 낮아졌다. 예를 들어 발생 빈도수에 따라 정보를 차등 취급하는 방법은 웹문서에 기술된 모든 내용에 거의 일률적인 중요성을 부여하는 전문색인 검색의 근본적인 단점을 보완하려는 움직임이다. 현재로서는 한 자원과 링크된 다른 자원의 수를 측정하여 순위적으로 결과를 제시하는 '구글검색' 알고리즘이 비교적 효율적인 전문색인 방법으로 인지되고 있다.

그러나 기존의 방법과 논리가 새로운 정보시대의 궁극적 대응책은 되지 못한다. 자원 내용에 관한 요약정보로 검색기능을 강화할 수 있는 메타데이터 시스템 개발에 국제적 관심이 집중되어 온 것도 이러한 현실을 반영하는 것이다. 그 중 특히 국제적 합의에 기반하여 제정된 '더블린코어' 메타데이터 스키마는 자원 유형과 무관하게 일반적으로 적용할 수 있는 국제 표준으로 이미 널리 알려져 있다. 그 밖에도 매체 유형의 특징을 반영한 메타데이터 스키마들이 다수 개발되어 구성요소가 단순한 '더블린코어' 메타데이터 스키마로 구현할 수 없는 상세 자원기술에 쓰이고 있다.

거대한 정보량과 그에 따른 다양한 메타데이터의 등장에 결부된 근본적인 과제는 교차 영역의 상호운용성 확보와 유지이다. 인터넷 망으로 연결될 수 있는 모든 정보에 상호운용성을 부여하려면 자원의 의미의 기계가독형 표현을 통한 웹 하부구조 개선 작업이 선행되어야 한다. 다시 말해서 현재의 웹을 자원의 단순한 군집이 아닌 의미의 결합체로 확장 발전시킴으로써 컴퓨터와 사람의 협력적 운용을 유도할 수 있어야 한다는 것이다. 이런 맥락에서 최근 몇년간 W3C가 추진해 온 시맨틱웹(Semantic Web)은 자원의 포괄적 표현과 자원 관계의 의미 있는 연결을 위한 논리적 규칙으로 정보학계가 주목해야 할 대상이다. 시맨틱웹은 웹 정보의 체계적이고 명확한 정의와 연결을 통해 효율적 검색, 통합, 재사용을 시도하는 새로운 기술로서 미래 정보 및 지식관리에 지대한 영향을 미칠 것으로 예측되기 때문이다.

이 논문의 목적은 이러한 시맨틱웹의 핵심 개념(URI, XML 네임스페이스)과

그 개념들을 활용한 시맨틱웹의 기반구조(RDF, RDF 스키마, DAML+OIL, OWL 마크업 언어)를 예제와 함께 살피고 시맨틱웹 기술의 적용으로 가능해지는 메타데이터의 구문적, 의미적 호환성 증가및 시맨틱웹의 활용방안을 논의하는 것이다.

2 시맨틱웹의 핵심개념

2.1 Uniform Resource Identifier (URI)

URI 는 자원에 부과되는 불변의 고유식별자를 지칭하는 URN (Uniform Resource Name)과 기존의 URL 양자를 모두 포괄하는 개념으로 사용되기도 하지만, 일반적으로 URI 체제는 불변성 식별자를 부과하는 체제의 개념으로 사용되고 있다. 자원에 할당되는 고유식별자의 불변성은 정보의 효율적 관리와 웹 자원 지식화의 초석이 되는 매우 중요한 특징으로서, 이러한 URI가 할당될 수 있는 자원의 예로는 다음과 같은 것들이 있다 (Manola and Miller 2002).

- 네트워크를 통한 접근이 가능한 것: 전자문서, 이미지, 정보서비스 (예: 오늘의 세계날씨에 대한 정보), 자원 컬렉션
- 네트워크를 통한 접근이 불가능한 것: 사람, 회사, 도서관의 일반장서
- 추상적 개념: '생성자', '주제', '표제'

2.2 XML 네임스페이스 (Namespace)

URI로 식별되는 이름들의 집합체인 XML 네임스페이스는 XML 문서의 요소명(element names)이나 속성명(attribute names)의 식별에 주로 사용되며, 다양한 기관에서 정의하는 요소명 또는 속성명들 사이의 잠재적 충돌가능성을 방지할 목적으로 제정되었다 (Bray, Holland, and Layman 1999). 네임스페이스의 용도는 다양하지만 대개 한 네임스페이스는 하나의 메타데이터 스키마에 포함된 모든 요소명의 집합을 대표하는 이름으로 활용됨으로써 어떤 기관이 자체 정의한 메타데이터 요소를 타 기관의 그것과 구별한다.

이하는 더블린코어가 사용하는 3개의 네임스페이스, 즉 1) 주요소의 식별을 위한 네임스페이스 (http://purl.org/dc/elements/1.1/), 2) 한정어에 적용되는 네임스페이스 (http://purl.org/dc/terms/), 및 3) 자료유형(Type)의 식별을 위한 네임스페이스 (http://purl.org/dc/dcmitype)의 예를 각각 나타낸 것이다 (Powell, et al, 2001).

1) 주요소의 네임스페이스에 각 요소명을 합하여 고유 URI를 생성하는 경우

- 더블린코어 '표제'의 URI: http://purl.org/dc/elements/1.1/title
- 더블린코어 '생성자'의 URI: http://purl.org/dc/elements/1.1/creator
- 더블린코어 '주제'의 URI: http://purl.org/dc/elements/1.1/subject
- 더블린코어 '자료유형'의 URI: http://purl.org/dc/elements/1.1/type

2) 한정어의 네임스페이스에 각 한정어명을 합하여 고유 URI를 생성하는 경우
- 더블린코어 '대체표제'의 URI: http://purl.org/dc/terms/alternative
- 더블린코어 '갱신일'의 URI: http://purl.org/dc/terms/modified
- 더블린코어 '초록'의 URI: http://purl.org/dc/terms/abstract

3) 자료유형의 네임스페이스에 각 자료유형의 통제어를 합하여 고유 URI를 생성하는 경우
- 자료유형 '이미지'의 URI: http://purl.org/dc/dcmitype/Image
- 자료유형 '컬렉션'의 URI: http://purl.org/dc/dcmitype/Collection
- 자료유형 '소프트웨어'의 URI: http://purl.org/dc/dcmitype/Software

자료유형의 인스턴스에 URI를 부과하는 것은 요소명에 URI를 할당하는 것과는 그 성격이 다르다. 예를 들어 더블린코어의 자료유형 요소명에 'http:/purl.org/dc/elements/1.1/type' 이라는 URI 가 할당되었고, 이 요소의 인스턴스에 해당하는 자료유형의 값에 URI를 할당하고 있는 점이 주목을 요한다. 자료유형의 값은 한 통제어의 값으로 볼 수 있으므로 동일한 '어휘'에 다른 의미를 부과하면서도 URI를 통해 충돌을 방지할 수 있다.

통제어에 URI를 부과하면 자원의 자유로운 재활용이 가능해진다는 점은 온톨로지 구축의 기반이 되며 따라서 웹 자원의 지식화 또한 가능해진다. 예로서 이미 구축되어 있는 어떤 시소러스에 URI 개념을 접목시킬 경우 웹 기반 '온톨로지'의 분산형 생성에 큰 발전이 있을 것으로 보여진다(인공지능 분야에서 '온톨로지'는 주로 '개념의 표시'로서, 기계가독형의 어휘 정의 및 어휘들 간의 관계 정의로 간주되고 있다. 그러나 이 논문에서는 이러한 일차적 정의 기능 외에 속성을 이용한 어휘의 통제 기능이 더해진 것으로 온톨로지 개념을 확장 사용할 것이다).

2.3 시맨틱웹 핵심개념의 활용

메타데이터의 상호운용성을 확보하는 관점에서 웹 정보 자원의 식별자가 구비해야 하는 가장 중요한 특성은 불변성이다. 즉 자원의 소재가 변경되더라도 식별자는 변하지 않아야 하는 것이다. 서버의 위치 정보에 근거하고 있기 때문에 자원의 소재 변경이 식별자의 수정으로 이어지는 URL은 따라서 현재 널리 활용되고는 있지만 웹 자원의 식별자로는 부적합하다고 간주되어 그 결과 제안된 것이 URI체제이다.

한번 자원의 식별자로 사용된 URI는 재사용될 수 없고, 한 자원의 새로운 버전이 생성되었을 경우에도 원본의 URI와는 다른 새로운 URI를 부과해야 한다. URI의 기능이 제대로 활용되기 위해서는 각 URI가 지칭하는 웹 자원으로의 연결을 담당하는 변환시스템이 필요하다. 즉

각 URI의 값을 실제 자원으로 연결되는 URL로 변환시켜 주는 시스템이 있어야 한다는 것이다. 이러한 URI체제를 통해 얻는 장점은 모든 링크를 URI 기반으로 변환함으로써 데드링크 현상이 최소화되도록 자원을 관리할 수 있다는 것이다. 즉 어떤 URI를 할당받은 특정 자원의 URL이 바뀐다고 하더라도 변환 시스템에 URL정보만 갱신하면 데드링크 없이 서비스가 지속될 수 있기 때문이다. 현재 URI 변환시스템으로 가장 널리 알려진 시스템은 미국 CNRI에서 개발한 핸들(HANDLE) 시스템이고, 더블린코어에서는 유사한 개념이지만 변환시스템에 의존하지 않는 영구적인 URL, PURL(Persistent URL)을 사용하고 있다.

URI 개념은 자원의 식별자로만이 아니라 메타데이터 속성 및 클래스의 식별자로도 활용되어야 한다. 세계 각국에서 정의되는 모든 메타데이터 속성은 고유의 URI로 구별될 수 있다. 따라서 아직 정의되지 않은 속성에 대해서만 새로운 URI를 부과하고 이미 정의된 메타데이터 속성은 다른 URI를 부과할 필요 없이 재사용함으로써 메타데이터 상호운영성 을 개선할 수 있을 것이다.

서로 연관성이 있는 URI 들은 한 네임스페이스에 속하게 된다. 예를 들어 현재 3개의 네임스페이스를 사용하고 있는 더블린코어 뿐만 아니라 잘 알려진 대부분의 메터데이터 스키마들도 네임스페이스를 사용하여 자체 요소들이 다른 기관에서 정의한 요소들과 충돌하지 않도록 URI를 관리하고 있다. 이러한 네임스페이스와 각 네임스페이스에 속한 URI를 등록할 수 있는 장치를 구축한다면 의미 충돌을 체계적으로 막고 새로운 스키마 설계 시에 기존의 정의를 사전 참조하는 일이 용이해질 것이다.

3 시맨틱웹의 핵심 기반구조

현 웹의 연장으로서의 시맨틱웹은 정보의 의미가 명확하게 정의된 환경, 즉 모든 자원 기술에 사용된 요소명과 요소명들 간의 관계, 속성명, 통제어와 통제어들 간의 관계 규명이 효율적으로 이루어져 컴퓨터와 사람의 협력이 용이해진 환경을 말한다. 이 시맨틱웹의 구축에 필요한 기술들은 점진적이며 상호 보완적인 발전을 거듭해왔다.

3.1 XML 스키마

3.1.1 XML 스키마 예제

XML DTD를 보완할 목적으로 제정된 XML 스키마는 시맨틱웹의 구성에 필요한 기반 구조를 구현하는 역할을 한다. XML 스키마의 장점은 네임스페이스 개념을 완벽하게 지원하고, 거의 모든 데이터 유형을 지원하며, 또한 카디날리티(cardinality) 조절이 용이하다는 것이다(Fallside 2001).

〈표 1〉 BookStore의 DTD

```
<!ELEMENT BookStore (Book)*>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

〈표 1〉의 DTD로 정의된 간단한 서적 관련 메타데이터 스키마를 XML 스키마 〈표 2〉로 변환시킨 후 그 이점을 비교 설명하고자 한다 (Costello 2002).

DTD의 데이터유형 지원이 약한 관계로 〈표 1〉에 제시된 모든 요소는 문자열 정의 'PCDATA'를 취했고, 카디랄니티 조절도 0,1,무한 등 3가지 중 선택하도록 규제되어 있다. 이에 비해 〈표 2〉의 XML 스키마에서는 새로운 네임스페이스를 지정할 수 있으므로 targetNamespace를 적용, 스키마 지정 문서에서 정의하고 있는 요소들이 http://www.books.org 라는 네임스페이스에 관련 저장되도록 하고 있다. 또한 ISBN과 Date 요소에 관한 정의가 보다 정교해졌고, 카디날리티 조절도 minOccurs와 maxOccurs 속성에 의지해 정수의 값 중에서 임의선택이 가능해졌음을 알 수 있다.

3.1.2 XML 스키마의 활용

메타데이터의 구조 표현에 있어서 DTD에는 많은 취약점이 있지만 그 중 가장 두드러진 것은 XML 네임스페이스 개념을 지원하지 않는 DTD 기반 메타데이터 표현에서는 다양한 메타데이터 스키마들을 혼용하는 형식의 자원 기술은 불가능하다는 점이다. XML 스키마가 개발된 목적의 하나도 이 네임스페이스 지원이라고 해야 할 것이다. XML 스키마를 사용하면 자원의 기술에 필요한 메타데이터 요소를 이미 선언된 네임스페이스에서 자유로이 취할수 있기 때문에 상세한 자원 기술이 용이해진다. 또한 XML 스키마는 메타데이터 요소의 인스턴스를 원활히 조절하고 다양한 데이터유형을 제공함으로써 속성의 값을 적절히 제어하는 기능을 제공한다.

3.2 Resource Description Framework (RDF)

상이한 메타데이터 도메인 간의 의미적 매핑(mapping) 지원이 상당히 미흡한 XML 스키마 만으로는 메타데이터 요소들의 의미와 다른 요소들과의 관계를 기계가독형으로 표현하는 일은 불가능하다.

〈표 2〉 BookStore DTD의 XML 스키마 변환

Bookstore XML 스키마

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace=http://www.books.org
    xmlns="http://www.books.org">
     <xsd:simpleType name="ISBNType">
          <xsd:restriction base="xsd:string">
               <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
               <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
               <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
          </xsd:restriction>
     </xsd:simpleType>
     <xsd:element name="BookStore">
          <xsd:complexType>
               <xsd:sequence>
                    <xsd:element name="Book"  maxOccurs="unbounded">
                         <xsd:complexType>
                              <xsd:sequence>
                                   <xsd:element name="Title" type="xsd:string"/>
                                   <xsd:element name="Author" type="xsd:string"/>
                                   <xsd:element name="Date" type="xsd:gYear"/>
                                   <xsd:element name="ISBN" type="ISBNType"/>
                                   <xsd:element name="Publisher" type="xsd:string"/>
                              </xsd:sequence>
                         </xsd:complexType>
                    </xsd:element>
               </xsd:sequence>
          </xsd:complexType>
     </xsd:element>
</xsd:schema>
```

이에 반해 RDF는 웹자원의 기술을 위한 일반용도의 언어로 제정된 것으로서, 그 장점 중 하나는 의미의 손상 없이 응용프로그램 간의 정보가 교환되도록 해당 정보를 표현하는 프래임워크를 제시한다는 점이다.

RDF는 레코드를 하나의 기술단위로 취급해온 기존 방식과는 달리 자원, 속성, 속성값을 하나의 단위로 취급하는 'TRIPLE'을 그 핵심개념으로 삼는다. 즉 RDF에서

는 1) 각 자원이 URI로 고유의 식별자를 갖고, 2) 자원을 기술하는 속성명이 고유한 URI로 표현되어 각 네임스페이스에서 정의된 속성을 사용함으로써 의미충돌을 방지하며, 3) 속성의 값으로 다른 URI가 지칭될 경우 이렇게 지정된 자원은 다시 기술 대상이 되어 그 자원에 대한 속성과 속성의 값을 새로 부과할 수 있다는 몇 가지의 강력한 특징이 있다. RDF의 정교한 자원 기술과 속성(predicate)에 의한 자원 관계의 자유로운 설정은 이러한 'TRIPLE' 개념에 기인한 것이다.

한 예로서 '홍길동' 이라는 이름의 교수가 변량분석(ANOVA)에 대한 강의안을 작성하였고 한국의 어떤 URI기관에서 이 교수에게 부여한 URI가 'http://uri.kr/professor/70032'라고 가정했을 때, 이 사실을 RDF로 표현한다면 다음과 같이 정리될 수 있다.

- Subject (주어): http://www.skku.ac.kr/statistics/anova.xml
- Predicate (술어): http://purl.org/dc/elements/1.1/creator
- Object (목적어): 홍길동 혹은 http://uri.kr/professor/70032

RDF에서는 자원을 '주어'로, 속성을 '서술어'로, 속성의 값을 '목적어'로 간주하여 표현하기도 한다. 이 경우 '주어'와 '술어'의 값은 반드시 URI를 사용해야 하지만, '목적어'의 값으로는 URI 또는 문자열의 사용이 모두 가능하다. 따라서 상기한 예의 주어, 술어, 목적어로 표현된 사실은 'http://www.skku.ac.kr/statistics/anova.xml' 자원의 생성자는 홍길동이다' 라는 문장이 된다. 주목할 점은 '자원 생성자'의 속성 표기에 더블린코어 요소인 creator의 URI를 사용하고 있다는 것, 그리고 효율적 인적자원관리를 위해 홍길동 교수에게 부여한 URI로 식별에서의 혼동이 완벽하게 방지되고 있다는 것이다.

또 만일 이 자원이 통계학강의 (http://www.skku.ac.kr/statistics/main.xml)의 한 부분이라고 가정했을 경우, 그 관계는 더블린코어의 'isPartOf' 요소를 사용하여 다음과 같이 명시할 수 있다.

- Subject: http://www.skku.ac.kr/statistics/anova.xml
- Predicate: http://purl.org/dc/terms/isPartOf
- Object: http://www.skku.ac.kr/statistics/main.xml

RDF는 이와 같은 방식으로 메타데이터를 인코딩, 교환, 재 사용할 수 있는 기반을 제공한다. 또한 메타데이터 의미의 상호 충돌 없는 표현에 필요한 구문구조를 갖추었고, XML 네임스페이스 개념을 완벽하게 지원한다. 표준화된 메타데이터들의 일관된 인코딩과 교환을 가능케 하는 RDF의 이러한 구조는 상이한 메타데이터 간의 호환성과 의미적 모듈화 (semantic modularity)를 제공하는 기반이 된다 (Miller 1998).

3.3 RDF 스키마

RDF 스키마는 메타데이터 스키마의

〈표 3〉 클래스 정의

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Description>

<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
</rdf:RDF>
```

속성(properties)과 클래스(classes)의 정의, 클래스와 클래스간의 관계, 속성과 속성간의 관계 등을 정의하는 언어로 제정되었다. XML 스키마는 다양한 데이터 유형의 지원과 인스턴스 조절에 큰 장점이 있고, RDF 스키마는 요소들의 의미 표현에 유리하기 때문에 양자는 상호 보완적인 기술로 이해되어야 할 것이다. 차후에는 이 두 스키마간의 긴밀한 협조도 예상되며 궁극적으로는 두 표준이 병합될 가능성이 많고, 이미 이들을 혼합해서 프로젝트에 적용한 사례도 발표되고 있다

(Garshol 2002).

RDF 스키마를 사용하면 사람이 이해하는 동시에 기계 처리가 가능한 형태로 메타데이터 속성과 클래스간의 관계를 정의할 수 있다. 이 체계적 표현의 핵심은 클래스와 속성의 명확한 정의, 속성들 간의 관계와 클래스들 간의 관계 정립이며, 체계적 어휘는 서로 다른 메타데이터들 사이의 구문적, 의미적 호환성을 촉진하는 기반이 되는 것이다.

3.3.1 '클래스'의 정의

RDF 스키마에서의 클래스는 일반적인 Type이나 Category 개념과 유사하며, 웹 페이지, 사람, 문서, 데이터베이스, 추상적 개념 등이 이에 속한다. 〈표 3〉에서는 클래스의 정의, 클래스와 클래스들 간의 상하위 관계가 RDF 스키마로 표현되어 RDF 스키마에서의 온톨로지 체계 확립 방법을 예시하고 있다.

예에서 RDF와 RDF 스키마에 관한 네임스페이스만 선언된 점으로 볼 때 RDF와 RDF 네임스페이스에 규정된 내용을 바탕으로 새로운 클래스나 속성이 정의될 것이라는 점을 짐작할 수 있다. rdf:Description은 바로 이러한 새 클래스나 속성의 기술 의도를 선언하고, rdf:ID에 정의할 클래스나 속성을 지정한다. 또 rdf:ID의 값으로 지정된 'MotorVehicle'은 rdf:type 속성을 사용하여 '클래스'를 상속하여 만들어진 '인스턴스 클래스'임을 정의하고, rdfs:subClassOf를 사용, 'MotorVehicle'을 자원(resource)의 하위클래스로 정의하였다. Truck, Van, MinVan 등은 모두 클래스로 (rdf:type 적용), PassengerVehicle, Truck 및 Van은 MotorVehicle의 하위클래스로, MinVan은 Van과 PassengerVehicle의 하위클래스로 각각 정의되고 있다.

3.3.2 '속성' 정의

RDF 스키마에서 '속성'에 관한 정의는 rdf:Property, rdfs:domain, rdfs:range, rdfs:subPropertyOf 등을 통해 이루어진다. 새로운 속성의 정의에는 rdf:type을 통한 '속성' 선언과 아울러 '속성'이 적용되는 클래스의 영역을 rdfs:domain으로, 속성이 취해야 하는 값의 범위를 rdfs:range로 각각 제한하는 작업이 따른다. 〈표 4〉는 이러한 속성 정의의 간단한 예이다.

이 예에서는 'registeredTo'를 '속성'으로 정의하고, 이 속성이 'MotorVehicle' 클래스에만 적용되는 것으로 한정하고, 'person' 클래스에 속한 것만이 이 속성의 값으로 올 수 있다고 규정하고 있다. 'rearSearLegRoom'의 경우도 먼저 '속성'으로 선언한 후, 이 속성은 'PassengerVehicle' 클래스에 속하는 것에만 적용되는 것이고 그 가능한 값으로는 정수(integer)를 한정하고 있다.

RDF 스키마에서는 또한 'rdfs:subPropertyOf'를 사용, 한 '속성'이 다른 속성의 하위속성인 사실을 표현할 수도 있는데, 이 때

〈표 4〉 속성 정의

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="registeredTo">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="http://www.example.org/classes#Person"/>
</rdf:Description>

<rdf:Description rdf:ID="rearSeatLegRoom">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Description>

</rdf:RDF>
```

속성에 적용되는 domain과 range는 그 하위속성에도 그대로 적용된다. 〈표 5〉는 이러한 속성간의 상·하위 관계를 정의한 예로서, 속성 'biologicalFather'를 이미 속성으로 정의되어 있는 'biologicalParent'의 하위 속성으로 규정하고 있다.

〈표 5〉 속성 및 하위속성 정의

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description rdf:ID="biologicalParent">
<rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>

<rdf:Description rdf:ID="biologicalFather">
  <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
<rdfs:subPropertyOf rdf:resource="#biologicalParent"/>
</rdf:Description>

</rdf:RDF>
```

〈표 6〉 RDF 스키마를 활용한 더블린코어 정의

```
<rdf:RDF

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms=http://purl.org/dc/terms/
    xmlns:dctype="http://purl.org/dc/dcmitype/">

  <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/relation">
          <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  </rdf:Property>

  <rdf:Property rdf:about="http://purl.org/dc/terms/isReferencedBy">
          <rdfs:subPropertyOfrdf:resource="http://purl.org/dc/elements/1.1/relation" />
          <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
  </rdf:Property>

</rdf:RDF>
```

한편 〈표 6〉은 더블린코어를 RDF 스키마로 정의하는 방법을 제시하는 예이다. 'relation'이 속성에 속하는 사실을 rdf:type 아닌 rdf:Property로 정의하였고 (축약구문), 이 요소가 정의되어 있는 URI를 언급하고 있다. 또한 'rdfs:subPropertyOf'를 이용, 'isReferencedBy'를 'relation'의 하위 속성의 하나인 속성으로 규정하고 있다.

4.3.3 RDF 스키마의 적용과 메타데이터 상호운용성

모든 메타데이터가 RDF의 triple로 표현되면 자원에 대한 기술이 주어-술어-목적어를 갖춘 논리적 단위로 저장되기 때문에 거대한 triple 지식창고를 생성하게 되는 효과를 가져온다. 이 지식창고를 활용하여 논리적 추론이 가능한 새로운 유형의 질의에 답할 수 있고 triple에 직접적으로 표현되어 있지 않은 암묵적 지식의 유추도 가능하게 된다. W3C에서는 triple을 시맨틱웹을 성취하는 핵심요소로 보고 있다.

또한 RDF(S)가 메타데이터 스키마 설계와 정의의 국제적인 기준으로 합의된다고 가정할 경우 모든 메타데이터와 연관된 클래스와 속성이 기계가독형으로 정의되고, 모든 속성의 적용영역과 범위가 domain 및 range로 명확히 제한될 수 있을 것이다. 이럴 경우 새로운 메타데이터 스키마를 정의하고자 하는 각 정보 기관들이 1) 가능한 한 이미 선언된 네임스페

〈표 7〉 DAML 네임스페이스 선언

```
<rdf:RDF
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml="http://www.w3.org/2001/10/daml+oil#"
  xmlns:xsd ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:dex ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex#"
  xmlns:exd ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex-dt#"
xmlns ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex#">
```

이스에서 메타데이터 요소들을 채택하고 2) 결여된 필요 요소들은 새로운 네임스페이스로 정의하되 기존 요소들과의 관계 설정이 가능한 부분은 RDF 스키마를 적용하여 정의한다면 메타데이터 스키마들 간의 상호운용성은 현재보다 크게 향상될 것이다.

4.4 DAML+OIL

시맨틱웹 구성에 본격적인 웹 온톨로지 언어가 필요한 이유는 메타데이터 스키마 정의의 근간이 되는 RDF 스키마에는 서로 동일한 의미의 요소, 역관계, union, intersection 등 메타데이터의 중요한 관계를 지원하는 능력이 결여되어 있기 때문이다. RDF와 RDF 스키마를 기반으로 이 두 언어에 부족한 모델링 요소를 확장, 강화하여 개발된 DAML+OIL은 웹자원을 기술하기 위한 시맨틱웹 마크업 언어이다 (Connolly et al. 2001). DAML+OIL의 네임스페이스 선언은 〈표 7〉과 같은 형태를 취한다.

DAML+OIL은 세상에 존재하는 모든 것을 객체(objects)로 분리하여 생각한다. 객체는 DAML '클래스', '속성', '데이터유형'으로 나뉘며, '데이터유형'은 XML 스키마에서 이미 정의된 것들로서 '클래스'의 정의에 사용된다. 다음은 DAML+OIL에서 강화된 클래스 및 속성 정의의 주요 골자를 각각 살펴본 것이다.

4.4.1 '클래스' 정의

〈표 8〉의 예는 동물 온톨로지의 개발과 관련한 DAML+OIL에서의 '클래스' 정의 구문 구조이다.

이 예에서는 'Animal'을 DAML 클래스로 우선 정의하고, 'Male'과 'Female'을 이 'Animal'의 하위클래스로 정의하면서 이들이 'disjoint'관계에 놓여 있음을 명시하고 있다. 즉 이 부분은 어느 누구도 'Male'인 동시에 'Female'이 될 수 없다는 사실을 규정한다.

4.4.2 '속성' 정의

DAML+OIL의 '속성'은 객체들 간의 관계를 기술하는 '객체형속성'과 객체를 데이터유형의 값과 연결시키는 '데이터형

〈표 8〉 DAML 클래스 정의

```
<daml:Class rdf:ID="Animal">
  <rdfs:label>Animal</rdfs:label>
  <rdfs:comment>
    This class of animals is illustrative of a number of ontological idioms.
  </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</daml:Class>

<daml:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <daml:disjointWith rdf:resource="#Male"/>
</daml:Class>
```

〈표 9〉 DAML 속성 정의

```
  <daml:ObjectProperty rdf:ID="hasParent">
    <rdfs:domain rdf:resource="#Animal"/>
    <rdfs:range rdf:resource="#Animal"/>
  </daml:ObjectProperty>

  <daml:ObjectProperty rdf:ID="hasFather">
    <rdfs:subPropertyOf rdf:resource="#hasParent"/>
    <rdfs:range rdf:resource="#Male"/>
  </daml:ObjectProperty>

  <daml:DatatypeProperty rdf:ID="shoesize">
  <rdf:type rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#decimal"/>
 </daml:DatatypeProperty>

 <daml:DatatypeProperty rdf:ID="age">
    <rdf:type rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
 </daml:DatatypeProperty>
```

속성'으로 나눠진다. 전자 daml:Object Property에 속하고, 후자는 daml:Datatype-Property에 속하는데, 〈표 9〉는 '속성'을 정의하는 방법에 관한 DAML+OIL 구문의 예이다.

이 예에서는 'hasParent'와 'hasFather'가 '객체형속성'으로 정의되는 동시에 이 중 'hasFather'는 'hasParent'의 하위속성(subPropertyOf)으로 나타나 있다. 또 'hasParent'가 적용될 수 있는 클래스는 'Animal'이며, 그 속성 값 역시 'Animal'에서 취할 수 있다고 명시하고 있다. '데이터속성'으로 정의된 'shoesize' 및 'age'를 'UniqueProperty'로 표현한 이유는 'shoesize'와 'age'는 유일하다는 뜻으로서, 그 중 'shoesize'는 소수를, 'age'는 정수를 그 값으로 취한다는 것을 알 수 있다.

4.4.3 '속성'을 제한하는 방법

클래스에 관한 제한 부여에는 'daml:Restriction', 'onProperty', 'toClass' 등을 사용한다. 가령 'Person'이라는 클래스를 'Animal'의 하위클래스로 규정한 후 'Person'의 'hasParent' 속성은 그 값을 반드시 'Animal' 아닌 'Person' 클래스에서 취한다고 명시하는 경우라면 〈표 10〉의 예와 같이 할 수 있다. 〈표 10〉에는 이외에도

〈표 10〉 'Person' 클래스의 정의

```
<daml:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasParent"/>
      <daml:toClass rdf:resource="#Person"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#hasFather"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#shoesize"/>
      <daml:minCardinality>1</daml:minCardinality>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

사람은 아버지가 한 사람(cardinality가 1)이어야 하고 신발 크기는 적어도 하나를 가져야 한다는 제한 조건(minCardinality)이 포함되어 있다.

4.4.4 DAML+OIL에서 OWL로의 발전

DAML이 RDF와 RDF 스키마에 부가한 데이터 모델링 표현력은 〈표 8〉-〈표 10〉의 예들이 제시하는 것 이외에도 상당히 다양하며 (Hamelen et al, 2001), 이를 기반으로 한 온톨로지와 웹서비스가 현재까지 매우 활발히 추진되어 왔다. 이러한 연구 사례 가운데 특히 주목할만한 것은 'wine (http://www.w3.org/TR/2002/WD-owl-guide-200021104/food.owl)' 과 'food (http://www.w3.org/TR/2002/WD-owl-guide-200021104/food.owl)' 개념을 대상으로 한 포괄적인 온톨로지 구축이라 할 것이다.

시맨틱웹 기술의 가장 최근 동향으로 제시되고 있는 Web Ontology Language (OWL)는 DAML 단계 온톨로지의 취약점으로 지적되어 온 용어 의미의 혼동을 보완한 웹 온톨로지 언어로서 DAML과는 거의 완벽한 수준의 호환성을 유지한다. 즉 OWL은 웹 문서 및 응용프로그램에 내재한 '클래스'와 '속성'들 간의 관계 정의 기능이 강화된 언어이다. 이하는 OWL을 정립하는 과정에서 발생한 주요 변동사항을 정리한 것이다.

- DAML 네임스페이스는 OWL 네임스페이스로 변환되었다 (http://www.w3.org/2002/07/owl).
- RDF와 RDF 스키마 워킹그룹에서 갱신한 내용이 OWL에 반영되었다: 즉 1) rdfs:domain과 rdfs:range가 중복 사용될 경우 교집합의 논리로 처리되며, 2) rdf:parseType="Collection"은 rdf:parseType="daml:collection"을 대체한다.
- DAML의 한정적 제한 기능은 도태되었다 (daml:cardinalityQ, daml:hasClassQ, daml:maxCardinalityQ, daml:minCardinalityQ).
- 의미의 일관성을 위해 DAML의 여러 클래스와 속성명들을 다음과 같이 개명한다.

DAML+OIL	OWL
daml:differentIndividualFrom	owl:differentFrom
daml:hasClass	owl:someValuesFrom
daml:toClass	owl:allValuesFrom
daml:UnambiguousProperty	owl:InverseFunctionalProperty
daml:UniqueProperty	owl:FunctionalProperty

- DAML이 결여한 대칭적 속성(SymetricProperty)을 첨가한다.
- DAML 요소 중 RDF, RDF 스키마와 동일한 의미를 갖고 있는 요소들을 삭제한다 (daml:comment, daml:domain, daml:range, daml:label, daml:isDefinedBy, daml:Literal, daml:Property, daml:seeAlso,

〈표 11〉 매운맛-레드소스-파스타-코스에 대한 정의

```
<owl:Class rdf:ID="PastaWithSpicyRedSauceCourse">
        <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#MealCourse" />
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasFood" />
                        <owl:allValuesFrom rdf:resource="#PastaWithSpicyRedSauce" />
                </owl:Restriction>
        </owl:intersectionOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#hasDrink" />
                        <owl:allValuesFrom>
                          <owl:Restriction>
<owl:onProperty rdf:resource="http://www.example.org/wine#hasColor" />
                              <owl:hasValue rdf:resource="#Red" />
                          </owl:Restriction>
                        </owl:allValuesFrom>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#hasDrink" />
                <owl:allValuesFrom>
                   <owl:Restriction>
<owl:onProperty rdf:resource="http://www.example.org/wine#hasBody" />
                        <owl:hasValue rdf:resource="#Full" />
                      </owl:Restriction>
                    </owl:allValuesFrom>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
        <owl:Restriction>
        <owl:onProperty rdf:resource="#hasDrink" />
        <owl:allValuesFrom>
                <owl:Restriction>
                <owl:onProperty rdf:resource="http://www.example.org/wine#hasFlavor" />
                <owl:hasValue rdf:resource="#Strong" />
                </owl:Restriction>
        </owl:allValuesFrom>
        </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
        <owl:Restriction>
                <owl:onProperty rdf:resource="#hasDrink" />
                <owl:allValuesFrom>
                     <owl:Restriction>
                <owl:onProperty rdf:resource="http://www.example.org/wine#hasSugar" />
                          <owl:hasValue rdf:resource="#Dry" />
                     </owl:Restriction>
                </owl:allValuesFrom>
        </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

daml:subClassOf, daml:subPropertyOf, daml:type, daml:value).

- DAML의 disjointWith 요소를 owl:unionOf, rdfs:subClassOf, owl:disjointWith 으로 대체한다.
- DAML의 equivalentTo 요소는 owl:sameAs 요소로 개명하고, 이 요소는 owl:sameClassAs, owl:samePropertyAs, owl:sameIndividualAs의 하위속성으로 취급하지 않는다.

4.5 OWL (Web Ontology Language)

4.5.1 OWL의 개괄

OWL은 웹 온톨로지에 연관된 지식창고(Knowledge Bases: 추론시스템에 축적된 논리적 명제의 집합)를 정의하는 언어이다. 명제에는 클래스의 구성원(members)들에 관한 사실들 뿐 아니라 온톨로지에 구문적으로 명확히 정의되지 않았으나 논리적으로 유추 가능한 사실들이 포함된다. 이러한 명제들은 하나의 온톨로지에 근거하거나 또는 다수의 분산형 온톨로지를 OWL 방식에 따라 수집한 온톨로지에 근거할 수 있다.

OWL 온톨로지는 클래스와 속성, 그에 적용되는 갖가지 제약사항(constraints)들의 집합으로서 다음과 같은 요소들을 포함한다 (Smith, McGuiness, Volz, and Welt, 2002).

- 클래스들 사이의 텍사노미 관계
- 데이터의 속성, 즉 클래스의 요소가 되는 속성의 값에 관한 기술
- 객체의 속성, 즉 클래스의 요소간의 관계에 관한 기술
- 클래스들의 인스턴스
- 속성들의 인스턴스

4.5.2 OWL 기반 온톨로지의 구축

만일 현존의 웹 에이전트에게 '식사 메뉴의 리스트를 제공하고 각 메뉴에 적합한 술을 추천하되, Sauterne은 제외시키라'는 요구사항을 준다면 이것은 실현 불가능한 질의로 판정될 것이다. 키워드 검색의 한계를 넘어서는 이러한 질의를 지원하려면 웹 자원 의미의 정확한 기술이 필요하다. 이하 한 음식의 코스에 대한 정의를 선언하고 있는 〈표 11〉 (http://www.w3.org/TR/2002/WD-owl-guide-20021104/food.owl) 을 통해 OWL 기반 온톨로지에 근거하여 가능해지는 새로운 정보서비스에 관해 살펴본다.

〈표 11〉에 따르면 '매운맛-레드소스-파스타코스'는 '식사코스'로 분류되고, '매운맛-레드소스-파스타'로 분류되는 구성원 중에서 주 음식이 제공되어야 한다. 또 그 제약사항으로서 밀도(body)가 높고(full) 맛이 강하며 당도가 낮은(dry) 적색와인을 음료로 제시하고 있다. 만일 이와 같은 OWL 방식으로 수많은 '식사코스'에 대한 정보를 표현하고 저장한다면 현재까지 어려웠던 다양하고 유익한 검색이 가능할 것이다. 즉 특정 '식사코스'에 어울리는 '와인' 리스트에 대한 상세정보,

또는 특별히 선호하는 '와인'과 어울리는 '식사코스' 등에 관한 검색이 문제 없이 수행될 수 있기 때문이다.

5 시맨틱웹 기술의 활용방안

이 논문에서 개괄한 시맨틱웹의 핵심개념과 기반구조는 미래의 정보서비스에 중요한 변혁을 가져 올 만한 특징을 가졌지만 시맨틱웹을 효율적으로 구성하는 작업에는 몇 가지의 정책과 관리체제가 병행되어야 한다.

5.1 네임스페이스 관리

체계적인 네임스페이스 관리는 현실적으로 가장 시급한 시맨틱웹의 과제 중 하나이다. 각 정보 기관들이 자체 네임스페이스 정책을 수립하고 이를 확실히 밝힐 필요가 있다. 주 네임스페이스를 결정할 뿐 아니라 새로 지정하는 네임스페이스에 대한 관리 방침을 또한 가지고 있어야 한다. 나아가 각 네임스페이스 관리기관에 대한 정보와 관련 메타데이터 요소들을 등록할 수 있는 국가적 메타데이터 등록기(Metada Registry)를 구축할 필요가 있다. 웹 상에서 전자사전과 같은 기능을 갖출 이러한 등록기는 다음과 같은 상황에서 참조할 수 있는 장치가 될 것이다 (오삼균 2002).

- 애플리케이션 설계에 필요한 메타데이터 스키마와 스키마 컴포넌트의 존재를 확인할 필요가 있는 경우와 해당 스키마를 확장하여 개발된 요소들에 대한 검색을 원하는 경우
- 메타데이터 설계자와 관리자가 메타데이터 요소의 정의, 용법 및 특정 요소가 취할 수 있는 값의 범위를 명확히 이해할 필요가 있는 경우
- 메타데이터의 요소와 요소값을 비교, 평가할 목적으로 특정 스키마, 요소, 통제어휘와 연계된 URI를 참조할 필요가 있는 경우
- 일반정보이용자가 메타데이터 어휘의 정의 또는 어휘의 적용상황에 관한 이해를 높여 효율적인 검색을 시도하고자 하는 경우

메타데이터 등록기는 메타데이터 스키마, 애플리케이션 프로파일, 그리고 통제어휘들을 선언하고 관리하는 효과적인 수단이 될 것이다. 이를 기반으로 메타데이터 스키마와 애플리케이션 프로파일들이 다수 생성되고 갱신되어 간다면 의미적이고 기계적인 상호운영성 유지를 위한 각 버전들 간의 관계는 체계적으로 유지될 수 있을 것으로 기대된다.

5.2 RDF(S) 기반 메타데이터 스키마의 등록체제

RDF(S) 기반으로 메타데이터 스키마가 정의되면 전 세계에서 사용하는 모든 메타데이터의 상호운영성은 크게 향상될 것이다. 모든 메타데이터 속성들 간의 상.하 관계가 명확히 정리되고, 각 메타데이터

스키마와 연관된 클래스들의 상.하 관계 또한 분명하게 정의할 수 있을 것이기 때문이다. RDF(S)를 인터페이스 언어로 채택하여 메타데이터 등록기가 구축될 경우에는 네임스페이스 및 관련 메타데이터 요소의 URI 관리 수준을 넘어서 메타데이터 '속성' 및 '클래스' 간의 관계에 대한 질의를 효율적으로 수용할 수 있는 바탕이 마련될 수 있다 (오삼균 2002).

다음은 모든 메타데이터가 RDF를 사용하여 기술되고 메타데이터 요소가 RDF(S) 기반으로 정의될 경우에 가능한 검색기능에 대한 간단한 기술이다.

- Triple 데이터베이스를 trace함으로써 많은 정보를 기계적으로 가공할 수 있게 되어 직접적인 관계가 기술되지 않은 자원에 대한 새로운 관계 설정과 이와 연관된 검색이 가능해진다.
- 고유의 URI로 표기된 자원들 간의 관계가 속성으로 기술되므로 다양하게 정의된 속성의 온톨로지 중 필요한 관계에 놓여 있는 자료의 수집이 용이해진다. 예를 들자면, 현 자원의 다른 버전(hasVersion), 딸림자료(hasPart), 대체자료(isReplacedBy), 다른 포맷의 같은 자료(hasFormat), 참고자료(references) 등을 용이하게 검색할 수 있게 될 것이다.
- 검색결과 자료가 지나치게 방대할 경우, 검색을 수행한 속성의 하위속성 중에서 협의의 검색을 시도하거나 하위클래스에 속하는 협의의 검색어로 재검색을 시도할 수 있다.

5.3 OWL 기반 온톨로지의 구축

과거의 인공지능, 지식표현 분야의 연구가 괄목할만한 결과를 내지 못한 가장 결정적인 이유는 온톨로지를 표현하는 표준 언어의 부재였다. 그러나 OWL은 시맨틱웹 연구자들이 공식 온톨로지 언어로 인정하고 있고 웹 환경에서 이에 기반한 분산형 온톨로지의 구축이 현실적으로 가능하기 때문에 웹을 지식화하는 도구로서 상당한 가능성을 시사하고 있다.

이하는 OWL 온톨로지를 기반으로 가능해진 다양한 검색기능의 유형 또는 현재 시도되고 있는 구체적인 질의들을 정리해본 내용이다.

- OWL의 'samePropertyAs'를 활용하면 속성명은 달라도 의미가 동일하기 때문에 정확성을 유지한 확장검색이 가능하고, 또한 'sameClassAs' 를 활용하면 같은 클래스에 속하는 구성원 중에서 취사선택한 어휘를 포함시켜서 확장검색을 시도할 수 있다.
- 'James Hendler'라는 이름의 '연구원'에 대한 정보 검색이 가능해진다. 흔히 현재의 웹 검색엔진으로 이 검색이 된다고 생각하기 쉽지만, '연구원' 신분에 대한 여과 없이 진행할 경우에는 '연구원' 아닌 'James Hendler'가 다수 검색될 가능성이 크다(Denker 2001).
- 'James Hendler'가 공저한 'SHOE'에 관한 논문을 참조한 다른 논문들의 검

색 또한 SHOE는 shoes와 다르다는 온톨로지 설정을 통해 용이해진다.
나아가 서지정보와 날짜의 구조를 알아야 하는 'James Hendler' 의 'SHOE' 에 관한 최근 논문의 검색 같은 질의도 가능해진다.
'결혼기념일' 식사코스로 적절한 메뉴 리스트를 제공하고 각 메뉴에 적합한 와인(wine)을 추천하되, Black Tower는 제외시켜라.
'매운맛의 파스타'와 함께 마시면 어울리는 '와인'을 추천하라.
특별히 선호하는 '와인'과 어울리는 '식사코스'를 검색하라.
'단맛이 강한 포도주'와 어울리는 '식사코스'에는 어떤 것들이 있는가?
주어진 일정 안에서 컨퍼런스에 관련된 여행 스케줄을 자동으로 처리해 줄 수 있는 서비스를 찾아라.
특정 신용카드를 받고 서울-뉴욕 왕복 항공권을 가장 저렴하게 판매하는 서비스를 찾아 예약하라.
웹에 있는 다양한 전시, 모임 정보 등을 개인정보시스템으로 용이하게 통합할 수 있는 서비스를 찾아라.
입사 지원자 A씨와 같이 일해 본 경험이 있는 사람을 검색하라.
지식표현 (Knowledge Representation) 분야에서 Java를 사용한 경력이 있는 사람을 검색하라.
내가 속한 회사에 관한 기사가 실리면 즉각 '보통사항'으로 통보하고, 만약 좋지 않은 측면에서 기사에 쓰였으면 '긴급사항'으로 통보하라.
- 표준기업 분류표에 근거한 경쟁사에 관련 기사가 나올 경우 즉각 통보하라.

6 결 론

이 논문에서 살펴본대로 점진적이고 부단한 발전 과정을 거쳐 온 시맨틱웹 관련 기술은 현재와 미래의 정보검색과 정보서비스에 다각도의 강력한 가능성을 제시하고 있다. 국내적으로는 새로 개발되는 메타데이터 스키마들의 네임스페이스 관리를 우선적으로 시도하여 국제 메타데이터 네임스페이스들과의 연계관계를 성립하고 국내 정보 시스템의 국제적 호환성을 증진시킬 필요가 있다. 또한 XML 스키마와 RDF 스키마의 포괄적인 활용으로 효율적인 메타데이터 스키마 설계를 도모하는 구체적인 움직임이 따라야 할 것이다. 시맨틱웹 연구의 주체가 되기 위해서는 XML/RDF 스키마의 구문구조와 OWL에 관한 실질적인 경험의 폭이 넓어져야 한다. 메타데이터 요소 간의 상호운용성을 초월하는 의미적 상호운용성을 확보함으로써 웹 자원의 지식화를 이룩하려는 시맨틱웹의 우선 과제는 주요 서비스 분야에서의 온톨로지 개발이라고 볼 수 있다. 이런 면에서 시소러스, 인공지능, 지식표현 분야등에서 축적된 연구결과는 효과적인 온톨로지 형성과 발전에 매우 요긴한

핵심 지식을 제공할만한 것들이다. 일례로 시소러스의 핵심지식은 OWL 온톨로지 구축에 필요한 기초지식이 될 수 있는 반면 속성 기술이 가능하고 클래스 정의에 속성으로 구체적 제약을 가할 수 있는 온톨로지는 시소러스에 담긴 의미의 모호성을 극복하는 좋은 방법이 될 수 있는 것이다.

참 고 문 헌

오삼균. 2002. 디지털도서관에서의 메타데이터의 역할. 『정보과학회지』, 20(8): 45-57.

Ahmed, K. et al. 2001. Professional XML Meta Data. Birmingham, UK: Wrox.

Berners-Lee, T. et al. 2000. "Semantic Web Development: Technical Proposal." 〈http://www.w3.org/2000/01/sw/DevelopmentProposal〉.

Bray, T., D. Hollander, and A. Layman. 1999. "Namespaces in XML." 〈http://www.w3.org/TR/REC-xml-names/〉.

Connolly, D., F. Hamelen, I. Horrocks, D. McGuinness, P. Patel-Schneider, L. Stein. 2001. "Annotated DAML+OIL Markup." 〈http://www.w3.org/TR/daml+oil-walkthru/〉.

Costello, R. 2002. "XML Schema Tutorial." 〈http://www.xfront.com/#schema〉.

Dean, M., D. Connolly, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, and L. Stein. 2002. "Web Ontology Language (OWL) Reference Version 1.0." 〈http://www.w3.org/TR/owl-ref/〉.

Denker, G., et al. 2001. Accessing Information and Services on the DAML-Enabled Web. Semantic Web Workshop 2001. Hongkong, China.

Fallside, D. 2001. "XML Schema Part 0: Primer." 〈http://www.w3.org/TR/xmlschema-0/〉.

Hamelen, F., I. Horrocks, and P. Patel-Schneider. 2001. "A Model-Theoretic Semantics for DAML+OIL (March 2001)." 〈http://www.w3.org/TR/daml+oil-model〉.

Manola, F. and E. Miller. 20002. "RDF Primer." 〈http://www.w3.org/TR/rdf-primer/〉.

Miller, E. 1998. "An introduction to the Resource Description Framework." 〈http://www.dlib.org/dlib/may98/miller/05mil

ler.html〉.

Pepper, S. and L. Garshol. (2002). The XML Papers: Lessons Learned on Applying Topic Maps. XML 2002 Conference and Exposition. December 8-13, 2002. Baltimore, Maryland. USA.

Powell, A., H. Wagner, S. Weibel, T. Baker, T. Matola, and E. Miller. 2001. "Namespace Policy for the Dublin Core Metadata Initiative (DCMI)." 〈http://dublincore.org/documents/2 001/10/26/dcmi-namespace/〉.

Smith, M., D. McGuinness, R. Volz, and C. Welty. 2002. "Web Ontology Language (OWL) Guide Version 1.0." 〈http://www.w3.org/TR/owl-guide/〉.

The Dublin Core Metadata Initiative. 1999. "Dublin Core Qualiers in RDF." 〈http://purl.org/dc/terms/〉.