디지털도서관에서의 메타데이터 역할

성균관대학교 오삼균

1. 서 론

정보량의 급격한 증가로 특징지어지는 인터넷시대는 자원의 생성 및 관리와 연관된 개념과 기술의 대변혁을 요구한다. 자원의 요약정보 또는 발달된 목록정보라고 볼 수 있는 메타데이터는 이 새로운 디지털 지식경제의 판도에서 중요한 역할을 담당하는 도구로서, 현재 가장 널리 알려진 더블린코어(Dublinc Core) 메타데이터 스키마를 비롯, 도메인의 성격과 정보 요구의 다양성, 또는 정보자원 자체의 유동적인 속성을 보다 면밀히 반영하고자 하는 새로운 유형들이 부단히 개발, 이용되고 있다. 메타데이터 간의 의미적 호환성 확보문제가 디지털도서관 구축의 관건으로 대두된 배경은 바로 이러한 스키마의 다변화에 있다.

이 논문의 관점은 메타데이터의 상호운용성이 구문체계의 기계가독형 표준화와 네임스페이스를 통한 스키마 관리, 그리고 발달된 메타데이터 등록기(Metadata Registry)의 사용으로 확보될 때 1) 최적화되고, 2) 자원의 상세기술 요구와 병존하며, 그 결과 3) 웹자원의 효율적인 지식화가 이루어질 수 있다는 것이다. 이런 맥락에서 메타데이터 설계의 기본원칙 (2장), 상호운영성 확보의 접근방식 (3장), 더블린코어 기반 메타데이터의 이용 (4장), RDF 핵심기술 (5장), 분산 메타데이터 사전 시멘틱웹과 메타데이터 등록기 (6장) 등 디지털도서관과 관련된 전반적인 관심사와 기술을 각각 살펴본다.

2. 메타데이터 스키마 설계의 기본원칙

듀발(Duval) 등[1]은 IEEE(Institute for Electrical and Electronics Engineers)의 LOM(Learnig Object Metadata) Working Group과 DCMI(Dublin Core Metadata Initiative)가 오타와 메타데이터 회의(2001년 8월)를 통해 수렴한 합의 내용을 기반으로 도메인과 무관하게 모든 메타데이터 스키마 설계와 애플리케이션에 적용될 수 있는 몇 가지 중요한 원칙을 제시하였다. 이하는 이 원칙을 발췌 정리, 또는 첨가 설명한 것이다.

2.1 모듈화(Modularity)

메타데이터의 모듈화는 자원의 출처나 관리방식 또는 기술 방법에 있어서 지극히 다양한 양상을 보이고 있는 디지털 환경에 일정한 체계를 부여하여 메타데이터의 재활용성과 상호운용성을 높이는 원리로서 네임스페이스를 그 핵심개념으로 한다.

모듈화된 메타데이터는 구문적, 의미적인 상호운용성을 유지하면서 상이한 스키마에 속한 메타데이터 속성과 클래스들을 조합할 수 있는 바탕으로써 매우 중요하다. 메타데이터 모듈의 원 설계자가 특정 조합을 염두에 두지 않았다 하더라도 애플리케이션 프로파일 설계 시에 다양한 모듈 혼합이 가능한 구조적 유연성, 즉 동일한 구문 표현 방식을 제공하기 때문이다. 예를 들어 XML과 같은 공통 구문 구조에 근거한 메타데이터 모듈들의 검색이나 관리는 개개 요소들의 기능을 강화하는 하나의 새로운 복합스키마로 통합될 수 있다.

웹 하부구조 (특히 XML)의 기본바탕이 되는 네임스페이스는 모듈화의 중요한 개념으로서, 일정 기관의 정책 또는 알고리즘에 따라 관리되는 어휘의 공식 집합으로 정의된다. 웹의 기본 프로토콜 HTTP나 지적창조물의 주제에 관한 규정에 따라 미국 국회도서관이 관리하고 있는 LCSH(Library of Congress Subject Headings) 등이 이네임스페이스의 예이고, 모든 메타데이터 요소세트 또한 그 유지기관의 규칙

과 관례에 의해 구속된 네임스페이스로 보아야 한다.

메타데이터 스키마 설계자는 한 어휘(메타데이터 요소)에 고유 정의를 부여하는 네임스페이스 선언과 그 선언의 다양한 집합을 통해 해당 메타데이터 요소들의 소속을 규명하고 식별이 가능하도록 한다(예: 더블린 코어 메타데이터 요소세트는 URI로 웹상에 정의되고 이 네임스페이스 선언의 적용범위내에 있는 모든 더블린코어 요소는 접두어 dc:로 구분한다).

2.2 확장성(Extensibility)

메타데이터 설계는 기본 스키마가 제공하는 상호운용성을 손상시키지 않으면서 애플리케이션의 특정 요구를 만족시키는 확장, 즉 새로운 요소의 추가를 허용해야 한다는 원칙이다. 대부분의 메타데이터 스키마에 적용되는 메타데이터 요소 (예 : 정보자원의 '생성자'나 '주제'의 개념)가 있는가 하면, 현실적으로 특정 도메인에만 적용되는 요소(예 : 고서의 '판심제')도 다수 나타나기 때문이다. 확장된 애플리케이션을 접하는 다른 애플리케이션은 호환성이 있는 부분을 제외한 이 애플리케이션의 기타 확장요소를 무시할 수 있어야 한다.

요소의 확장은 구체적으로 주요소 확장과 하위요소 확장으로 나누어 생각할 수 있는데, 주요소 확장의 경우, 확장된 요소에 대한 네임스페이스가 정의되어 있지 않으면 호환성을 상실한다는 점을 주지하고, 네임스페이스의 정의를 가능한 한 기계 가독형으로 하는 것이 바람직하다. 하위요소에 대한 확장도 네임스페이스에 정의할 것이 권장되나, 주요소의 네임스페이스 선언이 확보되면 확장을 무시할 경우 주요소의 값으로는 해석될 수 있으므로 의미의 호환성을 유지한다고 볼 수 있다.

2.3 상세성(Refinement)

메다데이터 설계는 도메인의 필요 또는 희망수준에 따라 최적의 상세성을 결정할 수 있는 구조이어야 한다는 것이 듀발 등이 제시한 또 하나의 원칙이다.

상세성의 첫째 개념은 기본 요소의 의미를 세분화하고 보다 분명히 하는 한정어의 추가다. 예를 들어, '삽화자(그린이)', '작자(글쓴이)', '작곡자', '조각자' 등은 '생성자'의 보다 명료한 유형이고, 자료의 '생성일', '갱신일', '인수일' 등은 모두 '자료' 속성의 보다 협의적인 용어로 쓰인다. 이러한 성격의 상세화는 유용성이 높고 또 해당 메타데이터 애플리케이션에서 매우 중요한 위치를 차지할 수도 있겠지만 상호운용성의 일반적 관점에서는 그 값을 보다 넓은 요소의 하위요소(subtypes) 로 간주해야 할 것이다

상세화의 둘째 개념은 일정 요소가 취할 수 있는 값의 범위를 규정하는 스킴 또는 가치집합(value set)의 명시이다. 요소의 값이 통제어(controlled vocabularies)로부터 선택되거나 특정 알고리즘에 따라 구성된다는 점, 즉 공통의 가치집합에 기반한다는 사실은 자동 프로세싱에서의 유용성과 애플리케이션 간의 의미적 호환성을 높인다. W3C의 [W3C-DTF]와 같은 날짜와 시간의 인코딩 표준은 이러한 예로서, 2002년 3월 6일과 같은 경우 지역에 따라 달리 해석될 수 있는 03/06/02 String 대신 2002-03-06으로 표기하여 모호한 메타데이터 표현을 해결하고 있다.

자원기술의 정확성을 높이는 통제어는 보다 발달된 주제기반 자원접근을 목표로 다양한 도메인에서 추구하는 지적 작업의 큰 비중을 차지하는 것으로서 특정도메인의 시소러스, 분류체계 등을 그 예로 들 수 있다. 메타데이터 컬렉션과 연관된 특정 통제어 규정은 애플리케이션이 일관된 검색과 브라우징 기능을 갖출 수 있는 기반이 된다.

2.4 다국어 지원(Multilingualism)

메타데이터 설계는 국제화(internationalization)와 지역화(localization)라는 다소 상반된 개념 간의 균형 유지로 정보자원의 이용을 극대화하려는 디지털 시대의 요구에 부응하는 작업이 되어야 한다는 원칙이다.

3. 상호운영성 확보의 접근방식

암스(Arms) 등 [2]은 다양한 기관들이 관리하는 기술적으로 상이한 구성요소들을 기반으로 정보이용자를 위한 일관된 서비스를 구축하는 작업을 디지털 도서관 상호운용성의 목표로 정의하고, 그 전제로서 참여기관들 간에 일어나는 세 단계의 합의를 설명하였다.

1) 메시지의 교환과 관계된 포멧, 프로토콜 및 안전장치(security systems)등의 기술적인 합의

2) 데이터와 메타데이터, 정보 해석(interpretation)에 관한 의미적 합의를 포함하는 켄텐츠(contents)의 합의

3) 정보접근, 콜레션과 서비스의 유지, 사용료 지급, 출처확인(authentication) 등의 조직적인 합의

상호운용성은 디지털도서관의 운영 및 서비스에 많은 영향을 미치는 근본적인 문제로서, 그 수준과 서비스 참여기관의 재정적, 인적 자원부담은 일종의 비례 관계에 놓여 있다. 즉 일반적으로 참여의 부담이 클수록 확보되는 상호운영성은 높아질 수 있다는 것이다.

전통적인 상호운영성의 접근방식은 모든 참여자들이 동일한 표준을 수용함으로써 통일된 방식으로 정보 서비스를 제공하는 것이다. 그러나 단일 표준양식의 포괄적인 적용에 의한 디지털도서관의 상호운용성 확보가 상당한 난제인 것은 경험적으로 이미 주지된 사실이며, 또한 유기적인 성격을 띈 구축 작업의 한 단면 만을 분리, 개선하여 만족할 만한 효과를 기대하기도 어렵다.

이하 NSDL(National SMETE Digital Library) 프로그램의 일환인 Site for Science 프로토타입[4]에서 제시된 개념을 기반으로 현존하는 상호운용성 확보의 세가지 유형을 정리하고, 웹자원의 효율적 지식화를 추구하는 새로운 대안으로써 "분산 메타데이터 사전"에 기반한 상호운영성 확보의 특징을 함께 논의한다.

3.1 기존의 방법

3.1.1 연합(Federation)에 의한 상호운용성

상호운용성의 전통적인 접근 방식이라 할 수 있는 연합은 참여기관들이 특정 기술조건(동일한 표준양식)에 맞추어 모든 정보서비스를 제공함으로써 상호운용성에 동참하는 형태를 취하는 것이다. Z39.50에 의한 온라인 목록 기록을 공유하는 도서관이나, 지구공간과학자료를 대상으로 한 산타바바라 주재 캘리포니아 대학의 ADEPT 프로젝트, 또는 NSDL (National SMETE Digital Library) 프로젝트의 일환으로 교육자료 콜렉션을 대상으로 하는 Smete.org 등이 이 연합의 예에 속한다. 연합 구성의 근본적인 난관은 참여기관들이 규정사항을 수용, 유지하기 위해 지대한 노력을 투자해야 한다는 사실이다. 또한 연합(Federation)에 기초한 디지털도서관에 새로운 서비스를 도입하는 과정은 모든 참여기관의 합의를 전제로 하기 때문에 많은 시간이 소모되는 단점이 있다.

3.1.2 수확(Harvesting)에 의한 상호운영성

거대한 연합의 구성에 따르는 부담을 고려하여 보다 느슨한 조직의 디지털도서관을 구성하려는 노력이 수확방법이다. 참여기관들은 포괄적인 합의보다는 적은 투자로 기본 서비스를 가능하게 하는 데이터 공유 방식의 합의에서 만족한다. 즉 메타데이터를 방출하는 구조에 관해 동의하는 것이다.

공유할 메타데이터 스키마에 관한 구조표현 형식으로는 현재 XML DTD나 XML 스키마가 널리 활용되고 있다. 그 예로 메타데이터 수확(metadata harvesting) 개념에 근거한 Open Archives Initiative (OAI) 프로토콜은 더블린코어를 상호운용성의 틀로 하여 DC 15개 요소에 관한 XML 스키마를 메타데이터 방출 표준으로 삼고 있다.

수확방법에 참여하는 개개의 디지털도서관들은 합의된 구조로 각 기관의 콜렉션에 관한 메타데이터를 생성하거나 이미 개발된 메타데이터를 변환하는 작업을 수행한다는 면에서 일치하지만, 그 이후의 양태에서는 차이를 보일 수 있다. 즉 메타데이터를 중앙저장소(Metadata Repository)로 방출하고 필요한 데이터를 인계받는 방식 또는 분산형 통합검색 방식 중 취택할 수 있기 때문이다. 정보검색이나 참조링크와 같은 각종 서비스는 참여기관의 필요에 따라 자의 생성되어 이용자에게 제공된다.

메타데이터 중앙저장소 사용의 강점은 모든 메타데이터가 단일 서버에 있기 때문에 일관성있는 서비스를 제공할 수 있다는 것이고, 약점은 데이터 갱신이 각 기관과 서버에서 동시 진행되지 않으면 데이터 무결성에 차질이 발생한다는 것이다. 메타데이터의 양이 상당히 방대할 경우에는 이러한 중앙집중식 방법 대신 분산형 통합검색을 시도하는 것이 더 적절할 수 있다. 즉 한 기관에서 표준 프로토콜을 사용하여 다른기관(들)에 질의를 보내고 합의된 양식(XML DTD 또는 XML 스키마)으로 검색 결과의 인스턴스 문서를 받아 이를 통합하는 방법을 쓰는 것이다. 그러나 이 방법은 많은 기관에서 실시간으로 메타데이터를 받을 경우 네트워크와 서버의 안정성에 따라 다른 검색결과를 낼 수도 있다.

수확(Harvesting)에 의한 방법 각 기관의 특성을 반영하는 상세정보가 표준 XML DTD나 XML 스키마에 수용되기 어렵기 때문에 결과적으로 모든 기관을 상대로 한 상세 검색을 거의 불가능하게 한다는 단점과 포맷에 관한 도메인간의 근본적인 합의가 전제되어야 하는 한계를 가지고 있다.

3.1.3 채집(Gathering)에 의한 상호운영성

여러 기관들이 공식적 협력 관계 아래 있지 않은 경우에도 기계 접근이 가능한 개방정보를 모음으로써 상호운용성의 기본 수준을 달성하는 방법으로, 웹 검색 엔진을 그 예로 들 수 있다. 콜렉션과 관계된 비용이 없어 많은 수의 디지털도서관을 포함하는 서비스를 제공할 수 있는 반면, 정보홍수 현상에 기인한 검색 효율의 저하, 비개방정보 사용불가와 같은 근본적인 취약점에 노출되어 있다.

3.2 "분산 메타데이터 사전"에 의한 상호운영성

메타데이터 설계의 가장 이상적인 형태는 도메인이 필요로 하는 상세 자원기술과 높은 수준의 데이터 호환성을 동시에 지원하는 것이다. 분산 메타데이터 사전에 의한 상호운용성은 메타데이터의 클래스와 속성 정의에 필요한 구문체계의 통일에 기반을 둔 상호운용성의 새로운 개념으로 정의할 수 있다. 메타데이터 요소의 표준화가 아닌 의미의 합의 기반(RDF, RDF 스키마)에 의존하기 때문에 사전에 포괄적인 합의가 이루어져야 하는 연합이나 공통요소에 의해 호환성을 추구하는 수확과는 근본적인 차이가 있는 이 방법의 골자는 아래와 같다.

1) 메타데이터 요소의 분산적 정의

해당 도메인이 요구하는 자원기술의 내용과 수준에 따라 메타데이터 요소를 정의하고, 그 자연적인 결과로 메타데이터 요소정의에 관한 '분산 메타데이터 사전'을 생성한다.

2) RDF 구문체계에 의거한 메타데이터 스키마

모든 메타데이터 스키마와 애플리케이션 프로파일의 설계는 기계가독형의 구문체계를 제공하는 RDF와 RDF 스키마에 기반하고, 설계된 스키마는 네임스페이스로 관리한다.

"분산 메타데이터 사전"에 의한 상호운영성은 메타데이터 설계의 자율성을 허락하고, 사람과 기계에 의해 동시에 이해되는 요소 정의를 구현함으로써 각 기관이 소유한 독특하고 상세한 정보의 공유를 가능하게 만드는 가장 발달된 형태의 상호운용성을 제공할 수 있다. 이와 관계된 RDF 기술, 메타데이터 등록기(Metadata Registry), DAML+OIL 온톨로지, 시맨틱웹 등의 문제는 각각 관련 장에서 언급될 것이다.

4. 더블린코어 기반 메타데이터의 이용

메타데이터의 세계적 이용상황을 보면 현재까지는 아직 더블린코어를 기반으로 상호운영성을 확보하는 서비스가 주조를 이루고, RDF 스키마를 사용하여 메타데이터를 정의하는 기관들이 점차로 증가하고 있는 추세이다. 그러나 Forrester 보고서[3]는 2004년을 기점으로 대다수의 기관에서 RDF 기술을 전격 수용, RDF가 메타데이터의 스키마를 정의하는 주도 기술이 될 것으로 전망하고 있다.

더블린코어 메타데이터의 두 가지 기본 목적, 즉 교차 도메인간의 자원 탐색(Cross-Domain Discovery)과 자원의 기술(Resource Description)[4]은 본질적으로 서로 역관계에 놓여 있는 개념이다. 기술의 수준과 상세성이 높아질수록 도메인 간의 의미적 합의는 더욱 어려워지기 때문이다.

1997년도부터 의미적 합의를 위한 15개의 기본 요소 외에 상세화 한정어 (refinement qualifier)와 스킴 한정어(scheme qualifier)의 사용을 지원하는 기능을 갖추기는 했지만 더블린코어는 여전히 자원의 단순 기술과 상이한 메타데이터 간의 최소 호환성을 확보하는 차원에서 적절한 도구로 간주되어야 할 것이다. 실제로 자원의 상세기술에 적합한 다양한 도메인의 메타데이터 스키마가 이미 다수 개발되어 있으므로 이들을 취사선택하는 한편 더블린코어를 교차 도메인간의 메타데이터 호환성을 이루는 주 도구로 활용하는 것이 매우 효율적인 방법일 수 있다.

4.1 상호운영성과 덤다운 원칙

더블린코어의 한정어 모델(qualification model)을 이용한 메타데이터 설계에서 유념해야 할 사항은 덤다운(dumbing-down) 원칙의 적용이다. 이 원칙은 예를 들어 더블린코어 기본 요소인 'Creator'의 한정어로 'Composer'를 썼을 경우 이 한정어의 의미를 이해하지 못하는 애플리케이션에서는 그 의미를 무시하고 기본 요소의 의미로 해석, 처리할 수 있어야 한다는 것이다. 그러나 이 원칙이 지켜질 수 없는 방식으로 임의 규정된 한정어를 사용할 경우 문제가 야기될 수 있다(그림 1).

그림 1의 예에서 최 상단 부분에는 해당 책의 저자가 'Alison Lurie' 이고, 그녀의 소속기관이 '코넬대학(Cornell University)'이라는 사실이 메타데이터로 표현되어 있고, 중간 부분에는 해당 책의 저자가 'Gary Cornell'이라는 사실이 또한 메타데이터로 표현되어 있다. 만일 '저자가 Cornell 인 자원을 검색하

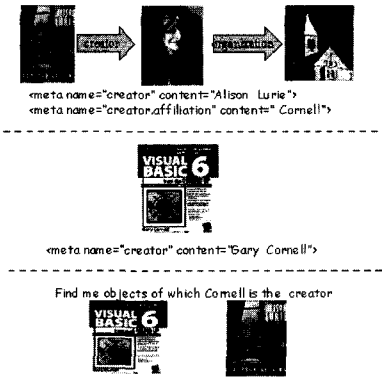

그림 1 덤다운 원칙 적용 시 잘못된 결과를 초래하는 예[4]

라'는 단순 질의를 받아 메타데이터 레코드에 덤다운 원칙을 적용하였다면 'creator.affiliation'의 소속기관이라는 의미를 무시하고 'creator', 즉 자원생성자로 처리하게 될 것이다. 즉 저자가 'Alison Lurie'인 책이 'Cornell'이라는 이름을 가진 사람의 책으로 잘못 검색되는 결과를 낳기 때문에 'creator.affiliation'이라는 한정어의 사용은 데이터의 변질과 의미적 호환성 차질의 요인이 된 것이다.

결론적으로 세분화 한정어 매커니즘은 더블린 코어의 기본적인 데이터 모델을 확장할 수 있는 유력한 수단이 될 수 있지만, 그 사용의 기본전제는 교차 도메인간의 자원 탐색 도구로서 더블린코어가 유지해야 하는 의미적 호환성의 핵이라고 할 덤다운 원칙의 무리없는 적용이다.

4.2 XML 인코딩 지침

메타데이터를 담는 컨테이너(container)로써 현재 가장 보편적으로 사용되고 있는 기술은 XML이다. 이하는 더블린코어 Architecture 그룹의 인코딩 권장지침서[5]를 정리한 것으로서, 표현의 일관성과 검색의 효율을 위해 모든 더블린코어 기반 메타데이터 설계와 인스턴스 문서 관리에 참조될 필요가 있는 내용이다.

- 권장사항 1 : 애플리케이션의 구현 기반으로는 XML DTD 보다 XML 스키마를 사용한다. XML 스키마를 기반으로 할 경우 스키마의 재사용이 용이하기 때문이다. 그러나, 어떤 경우에는 XML DTD와 스키마를 동시에 제공할 필요도 있을 것이고, 스키마가 없는 상황이라면 적어도 DTD는 제공되어야 한다.
- 권장사항 2 : 더블린코어 속성(properties)들의 독자적 식별을 위해 네임스페이스(namespaces)를 사용한다.
- 권장사항 3 : 더블린코어 속성은 XML 요소로, 그리고 속성의 값은 그 요소의 컨텐츠로 인코딩한다. (표제 표현의 예 : <dc:title>Korea/Japan World Cup</dc:title>)
- 권장사항 4 : 더블린코어의 15개 요소를 나타내는 속성의 이름은 모두 소문자로 표시한다. (예: 위의 예 중 dc:title이 소문자로 표시된 것)
- 권장사항 5 : 다중 속성값을 부가할 필요가 있을 때는 그 속성에 대한 XML 요소를 반복하는 형태로 인코딩한다. 예를 들어 두 표제의 표현은 아래와 같다.
  <dc:title>Gone with the Wind</dc:title>
  <dc:title>바람과 함께 사라지다</dc:title>
- 권장사항 6 : 상세화된 하위 요소들은 상위 요소들과 동일하게 취급한다. 예를 들어 날짜에 관한 상세화는 XML로 다음과 같이 표기한다.
  <dcterms:available>2002-06</dcterms:available>
  이 예에서 'dcterms'는 더블린코어 한정어의 네임스페이스를 의미하고 'available'이라는 요소가 'date (날짜)'의 하위요소라는 사실이 정의되어 있기 때문에 아래와 같은 인코딩은 쓰지 않는다.
  1. <dc:date refinement=available>2002-06 </dc:date> 또는
  2. <dc:date type=available>2002-06</dc:date> 또는
  3. <dc:date> <dcterms:available>2002-06 <dcterms:available> </dc:date>
- 권장사항 7 : 더블린코어 요소 값의 제어나 입력규칙의 표시를 위한 인코딩 스킴은 XML 요소의 'scheme'이라는 속성(attribute)을 사용하고, 인코딩 스킴의 이름은 그 속성의 값으로 표시한다.
  예 : <dc:identifier scheme=URI> http://www.uklon.ac..uk/</dc:identifier> (scheme은 identifier 요소의 속성 이름, URI는 인코딩 스킴의 값)
- 권장사항 8 : 더블린코어를 상세화할 경우 하위요소의 이름은 대,소문자를 혼합해서 쓸 수 있으나 항상 소문자로 시작한다. 인코딩 스킴의 이름 또한 대,소문자를 혼합해서 쓸 수 있으나 항상 대문자로

시작한다.

예: <dcterms:isPartOf scheme=URI>
http://www.bbc.co.uk</dcterms:isPartOf>
<dcterms:temporal scheme=Period>
name=The Great Depression; start=1929;
end=1939; </dcterms:temporal>

· 권장사항 9 : 어떤 요소 값의 언어를 표기하고자 할 때는 xml:lang이라는 속성을 사용해서 인코딩 한다.

예 : <dc:subject xml:lang=en>seafood
</dc:subject>
<dc:subject xml:lang=fr>fruits de mer
</dc:subject>

5. RDF 핵심기술

메타데이터의 인코딩, 교환, 재활용의 기반을 제공하기 위해 고안된 RDF는 메타데이터 클래스와 속성의 의미를 상호출돌 없이 표현하는 작업에 필요한 구문구조를 갖추고 있다. XML 네임스페이스 개념을 완벽하게 지원, 활용하는 RDF는 상이한 메타데이터 클래스 및 속성 간의 호환성과 의미적 모듈화(semantic modularity)를 제공할 수 있는 기반이 된다[6].

5.1 RDF 모델

RDF 모델은 자원을 기술하는 간단하지만 강력한 모델이다. RDF에 의해 기술되는 자원은 URI[7]를 통해 식별되고 속성을 가질 수 있다. 자원에 속한 속성들은 속성 타입(RDF:Type)으로 식별되며, 각 속성 타입은 그에 대응하는 값을 갖는다. RDF모델의 속성값은 더 이상 분화될 수 없는 최소단위의 값 (atomic value) 또는 다른 자원을 지칭하는 URI가 될 수 있다. 속성 값이 URI인 경우에는 속성이 두 자원 간의 관계를 표현한다(RDF모델에서는 URI로 지칭될 수 있는 있는 것을 모두 자원으로 간주함). 또한 속성의 값을 어떤 통제어 리스트에서 취하거나 표준규칙(예: W3CDTF)에 따르게 되는 경우가 있는데 RDF는 이러한 통제어나 표준규칙들은 클래스로 정의되며 그들 간의 관계를 표현하는 기반도 제공한다.

특정 자원에 속한 속성의 집합체를 그 자원에 대한 기술(RDF:Description)이라 하는데, RDF의 핵심은 자원과 그 자원에 대한 기술이 구문독립적(syntax-independent)인 모델이라는 점에 있다. RDF에서 가능한 자원 기술의 예를 보면 그림 2와 같다.

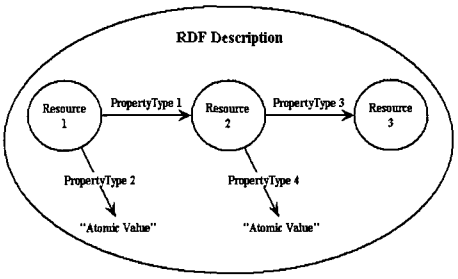

그림 2 RDF에서의 자원에 대한 기술

그림 2에서 보는 바와 같이 RDF 모델은 자원, 속성, 속성값의 3요소(triple 또는 statement)와 이들 요소간의 관계를 정의하는 방법을 통해 메타데이터의 의미(semantics)를 기계가독형으로 표현하는 기반을 제공한다. 이 점이 레코드단위로 자원을 기술하는 전통적인 자원기술방식과 RDF 방식 사이의 큰 차이이다. RDF는 한 자원을 하나의 기술단위로 취급하는 기능에 더해서 자원, 속성, 속성값 (주어, 술어, 목적어)의 형식으로 자원속성의 표현을 세분화하기 때문이다. 따라서 레코드 차원에서 만이 아니라 속성 차원에서의 관계 설정도 가능해지는 것이다.

다시 말해서 1) 각 자원이 URI로 고유의 식별자를 가진다는 점, 또한 2) 자원을 기술하는 속성명이 고유한 URI로 표현되어 일정 네임스페이스에서 정의된 속성을 사용하므로 의미충돌 없이 구분된다는 점, 3) 속성의 값으로 다른 URI가 지칭될 수 있고 속성의 값으로 지정된 자원은 기술 대상이 되며 따라서 그 자원에 대한 속성과 속성의 값을 새로이 지정할 수 있다는 점 (triple) 등이 RD모델이 제공하는 기술 방법의 강력한 점이다. 자원이 Triple로 기술되면 이 triple간의 관계는 기계가 용이하게 추적할 수 있는 기반이 된다(기계가독형).

예를 들어 '홍길동은 http://www.samsung.co.kr/life/ 의 자원 생성자다'를 RDF의 triple로 나타낸다면 그림 3과 같다.

주어 (자원)	http://www.samsung.co.kr/life/
술어 (속성)	Creator (저자)
목적어 (속성값)	홍길동

그림 3 RDF 모델 정의 1

또한 만약 '홍길동'과 관계된 URI로 지칭할 수 있는 자원이 있을 경우에 W3C의 RDF 모델 정의[16]에 따라 생성자를 더 상세히 표현한다면 그림 4와 같은 방향라벨선(directed label arc)이 될 것이다.

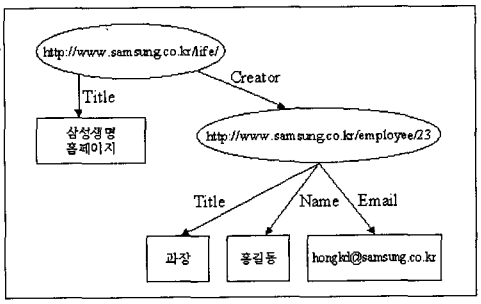

그림 4 RDF 모델 정의 2

그림 4는 그림 3의 홍길동이라는 속성값을 'http://www.samsung.co.kr/employee/23/'이란 식별번호를 가진 자원으로 대체, 자원에 대해 보다 상세한 표현이 가능해짐을 보여준다. 그림 4에서와 같은 URI 식별체계의 부여는 자원과 자원에 속한 속성들간의 연계를 용이하게 할 뿐 아니라 자원 기술(description)의 재사용을 도모하는 효율적인 방법이 될 수 있다.

RDF 인코딩의 적용은 메타데이터에 새로운 차원의 의미적 호환성을 부여하기 때문에 다양한 성격의 질의를 가능하게 한다. 예를 들어서 RDF를 지원하는 검색엔진에 'http://www.samsung.co.kr/life/를 자원으로 표현한 모든 속성들을 검색하라'고 요구한다면 이 자원을 기술하고 있는 속성들이 전세계를 대상으로 쉽게 검색될 것이다. 또한 'http://www.samsung.co.kr/life/를 속성의 값으로 사용하고 있는 triple을 검색하라'는 질의라든가 'http://www.samsung.co.kr/employee/23/를 자원의 URI로 사용하고 있는 속성의 리스트를 검색하라'는 질의도 가능하며, 나아가 'http://www.samsung.co.kr/life/와 isPartOf 관계를 가지고 있는 URI들을 검색하라'는 질의 또한 수용할 것이다. RDF가 이미 표준화된 더블린코어의 모든 relation의 하위요소로 표현된 자원들에 대한 연계관계 뿐만 아니라 새로 제정될 온톨로지(DAML+OIL)에 의한 풍부한 관계 설정을 가능하게 한다는 점은 RDF로 시맨틱웹의 기초를 다질 수 있는 중요한 이유이다.

5.2 RDF 구문체계(Syntax)

모델 차원에서는 데이터의 인코딩, 전송, 파일로의 저장 등과 같은 문제를 해결할 수 없기 때문에 RDF를 사용하여 응용 프로그램간의 커뮤니케이션을 지원하는 것은 RDF 구문체계(syntax)이다. W3C의 RDF 구문체계 표준서[8]는 구문체계를 단일 규정하지 않고 XML을 포함하여 RDF 모델의 표현이 가능한 기타의 구문체계를 허용하고 있으나, 현재 RDF를 담을 수 있는 최적의 컨테이너로 통용되고 있는 것은 XML이다[9]. RDF의 중요한 목표는 XML을 통해 메타데이터 구문구조를 표준화하고 의미(semantics) 표현 방식에 기준하여 메타데이터의 호환성을 확보하자는 것이다. 이런 점에서 XML과 RDF는 상호보완적이라고 할 수 있다.

상이한 메타데이터를 이용하여 주어진 응용 프로그램의 문제 영역(problem domain)에 적합하게 표현하려면 메타데이터 요소들 간의 의미충돌을 방지하는 방법이 필요하다. 예를 들어, 더블린코어의 'title' 요소는 표제, 즉 자원에 주어진 이름으로 정의되어 있는데, 만일 다른 어떤 메타데이터에서 'title'이라는 동명 요소를 조직 내에서의 직급으로 정의하고 있다면, 이 메타데이터 집합과 더블린코어기반 메타데이터 집합 사이에는 'title' 요소에 대한 의미의 구분이 필요해질 것이다. 이 경우 XML 네임스페이스는 더블린코어의 어휘를 정의한 스키마를 지정할 수 있는 방법을 제공함으로써 동명이의 요소 간의 충돌을 막는다. 다시 말해서 더블린코어의 'title'은 다른 기관에서 정의한 'title'과는 상이한 URI를 가지면 되기 때문이다. 그림 4의 예를 더블린코어를 사용하여 RDF/XML로 표현하면 예제 1과 같다.

예제 1에서 동명요소 'title'은 표제(더블린코어)와 직급(삼성생명 메타데이터)으로 각각 의미의 충돌 없이 정의되고 있다. 이와 같이 RDF 구문체계에서는 XML 네임스페이스를 이용하여 다종의 메타데이터들 사이의 의미충돌을 방지하고, 이종의 메타데이터(vCard)로 한 메타데이터 집합에 속한 요소(dc:creator)를 응용 프로그램의 문제 영역에 가장 적합하게 상세기술(refinement)하는 기능이 제공된다.

5.3 RDF 스키마

RDF 데이터 모델은 속성과 속성값을 이용한 자원간의 관계기술을 지원하는 반면 이러한 속성과 클래스의 정의, 클래스와 클래스 간의 관계, 속성과 속성간의 관계 등을 설정할 수 있는 방법은 제공하지 않

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
   syntax-ns#"
   xmlns:dc=http://purl.org/metadata/dublin_core#
   xmlns:vCard = "http://www.w3.org/2001/vcard-rdf/3.0#
   xmlns:sl = http://www.samsung.co.kr/metadata/1.0#>
   <rdf:Description about="http://www.samsung.co.kr
   /life/">
     <dc:title>삼성생명 홈페이지</dc:title>
       <dc:creator>
         <rdf:Description about=http://www.samsung.
          co.kr/employee/23>
            <vCard:Name>홍길동</vCard:Name>
            <sl:title>과장</sl:title>
            <vCard:Email>hongkd@Samsung.co.kr
            </vCard:Email>
         </rdf:Description>
       </dc:creator>
    </rdf:Description>
</rdf:RDF>
```

예제 1 그림 4의 도식화에 대한 RDF/XML 표현

는다. 이러한 기능을 지원하는 것은 RDF 스키마 정의이다. RDF 스키마를 통해 사람이 이해할 수 있고(human-readable) 또한 기계가 처리할 수 있는(machine-processable) 메타데이터 속성들 간의 관계를 체계적으로 표현할 수 있다. 체계화의 핵심은 메타데이터 스키마 설정에 있어서 클래스(class)와 속성(property)의 명확한 정의 및 클래스와 속성 간의 관계 정립에 있으며, 이와 같이 체계화된 어휘는 이종 메타데이터의 교환, 사용 및 확장을 촉진할 것이다.

RDF 스키마 표준문서[10]에 따르면 RDF 스키마는 기본적으로 RDF 자원들의 모음(collection)으로 표현된다. RDF 스키마의 핵심 용어(vocabulary)는 http://www.w3.org/2000/01/rd-schema#의 URI로, 핵심 RDF 네임스페이스는 http://www.w3.org/1999/02/22-rdf-syntax-ns#의 URI로 식별되어 있다. 표준 문서는 총 14개의 클래스(class)와 11개의 속성들에 대한 정의를 포함하며, 각각의 경우에 관한 설명과 예를 보여준다.

이하 두 예제로 RDF 스키마를 통한 클래스와 속성의 정의 방법과 정의된 RDF 스키마를 RDF 인스턴스 문서 생성에 이용하는 방법에 관해 각각 알아본다.

'Person' 클래스가 http://www.w3.org/2000/03/examples/classes 안에 정의된 'Animal' 클래스의 'subclass'임을 보이고 있는 예제 2는 객체지향형 설

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

<rdfs:Class rdf:ID="Person">
  <rdfs:comment>The class of people.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/
   2000/03/example/classes#Animal"/>
</rdfs:Class>

<rdf:Property ID="maritalStatus">
  <rdfs:range rdf:resource="#MaritalStatus"/>
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property ID="ssn">
  <rdfs:comment>Social Security Number</rdfs:comment>
  <rdfs:range rdf:resource="http://www.w3.org/2000/03/
   example/classes#Integer"/>
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property ID="age">
  <rdfs:range rdf:resource="http://www.w3.org/2000/03/
   example/classes#Integer"/>
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>

<rdfs:Class rdf:ID="MaritalStatus"/>
<MaritalStatus rdf:ID="Married"/>
<MaritalStatus rdf:ID="Divorced"/>
<MaritalStatus rdf:ID="Single"/>
<MaritalStatus rdf:ID="Widowed"/>

</rdf:RDF>
```

예제 2 Person 스키마

계의 클래스상속(Inheritance) 개념을 이용해 도메인 내/외부에 정의된 클래스를 현 도메인이 요구하는 의미에 맞게 정의할 수 있음을 보이는 예이다. 이 'Person' 클래스에는 또한 3개의 속성(Property), 즉 'maritalStatus', 'ssn', 'age'이 정의되어 있고, 모든 속성이 가질 수 있는 값의 범위는 'rdfs:range', 그 속성이 적용되는 클래스의 규명에는 'rdfs:domain'을 이용하고 있다.

이 예제 2의 'Person' 스키마 문서가 http://www.skorea.org/RDF-schema/의 URI로 식별된다고 가정할 경우 생성될 수 있는 RDF 인스턴스 문서는 예제 3과 같다.

```
<? Xml version=1.0?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-
                    syntax-ns#
         xmlns:rdfs=http://www.w3.org/2000/01/rdf-schem
                    a#
         xmlns:extNS=http://www.w3.org/2000/03/
                    example/classes#
         xmlns:s=http://www.skorea.com/RDF-schema#>

<rdf:Description rdf:about=http://www.skorea.org/JohnDoe>
          <rdf:type resource=http://www.skorea.com/
                              RDF-schema#Person/>
          <s:maritalStatus>
               <rdf:value>Single</rdf:value>
          </s:maritalStatus>
          <s:ssn>
              <extNS:Integer><rdf:value>12334
              </rdf:value></extNS:Integer>
          </s:ssn>
          <s:age>
             <extNS:Integer><rdf:value>35
             </rdf:value><extNS:Integer>
          </s:age>
</rdf:Description>
```

예제 3 Person 스키마의 RDF 인스턴스 문서

5.4 RDF 기반 메타데이터 네임스페이스와 애플리케이션 프로파일

XML 네임스페이스(Namespace) 스키마는 한 네임스페이스에서 정의된 모든 요소 이름들의 집합체로서 이 집합체의 각 멤버는 고유의 URI를 가진다. 따라서 네임스페이스는 메타데이터 요소들을 식별하고 애플리케이션 사이에서 의미의 혼동 없이 사용하는 중요한 수단이 된다[11].

애플리케이션 프로파일(Application Profile)은 하나 이상의 네임스페이스로부터 추출한 메타데이터 요소로 새로 구성된 메타데이터 스키마로서, 특정 애플리케이션을 위해 최적화되고, 새로운 요소를 정의할 수는 없으나 허용된 스킴과 값을 정의할 수는 있다 [12]. 애플리케이션 프로파일의 표현에는 현재 RDF 스키마 또는 XML 스키마가 쓰이는데, RDF 스키마에 근거한 대표적 애플리케이션 프로파일들은 SCHEMAS 프로젝트[13]에, XML 스키마에 근거한 애플리케이션 프로파일의 생성은 Hunter [14]에 각각 잘 예시되어 있다(상세한 리스트는 부록을 참조할 것).

RDF 기반으로 정의된 더블린코어의 예

이하의 예제들은 더블린코어(DC)와 더블린코어한정어(DCQ)의 요소, 스킴에 대한 RDF 인코딩 방식을 세 부분으로 나누어 표현한 것이다. 이 예제들의 요소 정의는 EOR(Extensible Open Rdf)에 근거하고 있다[15].

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf
                    -syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-sche
                    ma#" xmlns:dc="http://purl.org/
                    dc/elements/1.1/"xmlns:dcq="http://purl.
                    org/dc/terms/"
         xmlns:eor="http://dublincore.org/2000/03/13/
                    eor#">
   <!-- Description of Schema -->
  <eor:Schema rdf:about="http://purl.org/dc/terms/">
  <rdf:value>The Dublin Core Element Set Qualifier
   Vocabulary</rdf:value>
  <dc:title>The Dublin Core Element Set Qualifier
   Vocabulary</dc:title>
  <dc:publisher>The Dublin Core Metadata
   Initiative</dc:publisher>
  <dc:description>The Dublin Core Element Set Qualifier
   Vocabulary is an richer vocabulary intended to
   facilitate discovery of resources.</dc:description>
  <dc:language>English</dc:language>
     <dc:relation rdf:resource="http://dublincore.org/
      documents/dcmes-qualifiers/" />
     <dc:date>2000-07-11</dc:date>
   </eor:Schema>
   ..
</rdf:RDF>
```

예제 4 DC/DCQ RDF의 요소/한정어에 대한 RDF 스키마 인코딩 선언부

예제 4는 현재 문서가 참조하고 있는 네임스페이스에 대한 선언부분을 포함하며, RDF로 표현하고 있는 문서 자체에 대한 정보를 <eor:Schema> 안에 DC의 'title', 'publisher', 'description', 'language', 'relation', 'date' 등의 요소를 사용하여 기술하고 있다. 즉 <eor:Schema>안에 표현된 내용은 RDF 스키마를 사용하여 DCQ의 어휘를 정의한 것이다. 두 번째 부분은 예제 5와 같다.

예제 5에서 모든 요소나 한정어는 'rdf:Property'를 사용하여 정의되고 있다. 'rdf:about'의 값으로는 요소나 한정어의 공식명칭이 URI의 값으로 표기되고, 'rdfs:label'에는 사람이 이해할 수 있도록 그 요소나 한정어의 이름을 표기하며, 'rdfs:comment'를 통해서 용어에 대한 설명을 기술한다. 상위 요소가 존재할 경우 'subPropertyOf'를 통해 상위 요소의 URI를 표

기하고 (alternative가 title의 하위요소라는 것이 정의됨), 'isDefinedBy'를 사용하여 해당 용어의 정의를 기술하고 있는 문건의 URI를 표시한다. DC/DCQ에 해당하는 요소와 한정어는 모두 이와 같은 패턴으로 표시되며, 더블린코어가 아닌 다른 메타데이터의 경우에도 동일한 방식으로 메타데이터 속성(property)들 간의 상하위 관계를 표기할 수 있다. 마지막으로 예제 6은 스킴을 표현하는 부분이다.

'Date'에 관한 인코딩 스킴을 규정하고 있는 예제 6에서 DateScheme의 하위요소 스킴에 대한 표현에는 'rdf:Description'을 통해 기술대상 요소의 URI와 스킴의 해당범위가 우선적으로 지정되어 있다. 실제 스킴에 해당하는 부분을 기술하는 Class에서는 요소 표현과 유사하게 'rdfs:label'로 요소명에 대한 사람의 이해를 돕고, 'rdfs:comment'를 통해 해당 스킴을 설명하고 있으며, 상위 클래스에 대한 표기(subClassOf), 참조문건의 위치 정보(seeAlso), 정의된 문건의 위치정보(isDefinedBy) 등을 나타내고 있다.

5.5 RDF 스키마와 XML 스키마 비교

애플리케이션 프로파일을 정의하기 위한 스키마로는 RDF스키마와 XML 스키마가 있으며, 그 각각의 장단점은 아래와 같다. (네임스페이스를 지원하지 않는 XML DTD는 논의의 대상에서 제외한다.)

RDF 스키마는 메타데이터 요소의 풍부한 의미 기술에 탁월한 반면 데이터의 구조표현, 인스턴스에 관한 카디널리티(cardinality), 데이터의 유형 제약 측면에서 문제가 있어 실제 시스템 제어에 필요한 정

```
<rdf:Property rdf:about="http://purl.org/dc/terms/alternative">
    <rdfs:label>Alternative</rdfs:label>
    <rdfs:comment>Any form of the title used as a substitute or alternative to the formal title of the resource.</rdfs:comment>
    <rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/title" />
    <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
</rdf:Property>
```

예제 5 DC/DCQ의 요소, 한정에 대한 표현 RDF 스키마의 정의

```
    <!-- Date range declarations -->
    <rdf:Description rdf:about="http://purl.org/dc/elements/1.1/date">
        <rdfs:range rdf:resource="http://purl.org/dc/terms/DateScheme" />
    </rdf:Description>
    <!-- Encoding Schemes -->
    <rdfs:Class rdf:about="http://purl.org/dc/terms/DateScheme">
        <rdfs:label>Date Encoding Schemes</rdfs:label>
        <rdfs:comment>A set of date encoding schemes and/or formats</rdfs:comment>
        <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
    </rdfs:Class>
    <rdfs:Class rdf:about="http://purl.org/dc/terms/W3CDTF">
        <rdfs:label>W3C-DTF</rdfs:label>
        <rdfs:comment>W3C Encoding rules for dates and times - a profile based on ISO8601</rdfs:comment>
        <rdfs:subClassOf rdf:resource="http://purl.org/dc/terms/DateScheme" />
        <rdfs:seeAlso rdf:resource="http://www.w3.org/TR/NOTE-datetime" />
        <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
    </rdfs:Class>
    <rdfs:Class rdf:about="http://purl.org/dc/terms/Period">
        <rdfs:label>DCMI Period</rdfs:label>
        <rdfs:comment>A specification of the limits of a time interval.</rdfs:comment>
        <rdfs:subClassOf rdf:resource="http://purl.org/dc/terms/DateScheme" />
        <rdfs:seeAlso rdf:resource="http://purl.org/dc/documents/rec-dcmiperiod-20000619.htm" />
        <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" />
    </rdfs:Class>
```

예제 6 DC/DCQ의 스킴에 대한 RDF 스키마 인코딩

보의 표현에는 한계가 있다. XML 스키마는 데이터의 구조표현, 카디날리티, 데이터 유형에 대한 제약 등을 효과적으로 지원하지만, 메타데이타 도메인간의 동적인 매핑(mapping)을 가능하게 하는 의미정보의 표현에는 약하다. RDF 스키마는 특정도메인의 메타데이터 모델에 관한 의미정보의 표현에 이상적이고, XML 스키마는 시스템 구축의 구체적인 제어를 표현하는 작업에 효과적이라고 할 수 있다.

RDF 스키마와 XML 스키마의 장단점은 이렇게 서로 대치되며 결과적으로 상호보완적이다. 따라서 이 두 스키마의 결합이 효율적인 접근이 될 수 있겠는데, 구체적으로는 1) XML 스키마의 annotation 타입 안에 RDF 스키마의 'Class/subClassof', 'Property/subPropertyOf'를 끼워 넣는 방식과; 2) 외부 RDF 스키마 파일로의 링크를 XML 스키마에 포함하는 방식을 고려할 수 있다[13].

6. '분산 메타데이터 사전 시맨틱웹'과 메타데이터 등록기

시맨틱웹은 기계와 인간이 동시에 의미 혼선 없이 전자정보를 교환할 수 있는 협동적인 웹으로서 W3C에서는 그 구현을 활동 목표 중의 하나로 삼고 있다[17].

웹자원의 지식화를 가능하게 할 시맨틱웹의 실현에는 메타데이터 요소들의 의미가 웹 애플리케이션간에 자동 공유된다는 주요 전제가 따른다. 따라서 현재 유일하게 메타데이터의 클래스와 속성에 관한 기계가독형의 구문체계를 제공하는 RDF(S)의 일괄적인 사용이 시멘틱웹 하부구조 형성에 큰 관건이다.

시맨틱웹 구현의 또 다른 하부구조를 형성하고 있는 메타데이터 등록기(Metadata Registry:MDR)는 디지털도서관 연구의 중요한 한 분야로서, 다양한 정보 요구에 부응하는 메타데이터와 애플리케이션 프로파일의 스키마가 증가함에 따라 MDR과 그 내용에 관한 관리의 중요성도 크게 부각되고 있다[]. MDR은 1) 이미 개발된 메타데이터 스키마와 애플리케이션 프로파일들의 용이한 식별과 참조, 2) 상이한 메타데이터 스키마들 간의 자동 매핑, 3) 각 메타데이터 요소가 취할 수 있는 값의 통제어휘 연결 등의 중요한 도구가 될 수 있을 것으로 기대된다. 일종의 전자사전과 같은 기능을 갖춘 MDR은 다음과 같은 상황에서의 참조가 가능할 것이다.

· 애플리케이션 설계에 필요한 메타데이터 스키마와 스키마 컴포넌트의 존재를 확인할 필요가 있는 경우 및 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 통해 검색의 효율을 높이려 하는 경우

· MDR은 메타데이터 스키마, 애플리케이션 프로파일, 그리고 통제어휘들을 선언하고 관리하는 효과적인 수단으로서 메타데이터 스키마와 애플리케이션 프로파일들이 다수 생성, 갱신되어감에 따라 의미적이고 기계적인 상호운영성의 확보를 위해 각 버전들간의 관계를 유지해 나갈 수 있을 것이다.

7. 결 론

메타데이터의 적용으로 보다 나은 서비스를 제공하는 디지털도서관을 구축하기 위해서는 메타데이터의 공유된 설계원칙들과 타 기관을 대상으로 한 호환성의 수준을 유기적으로 고려해야 한다.

분야별로 현격한 차이를 보일 수 있는 이용자의 요구를 합리적으로 반영하는 메타데이터 스키마에 관한 연구와 함께 기존 메타데이터 스키마와 애플리케이션 프로파일들에 대한 철저한 조사를 선행해야 할 것이다. 이 때 기존 요소 중 적용가능한 부분은 재사용하는 것이 상호운영성면에서 매우 바람직한 선택으로 보인다. 이미 매체별로 개발된 다수의 메타데이터 스키마가 각 관련 집단의 지속적인 노력을 거쳐 건설적으로 유지, 발전되고 있다는 사실을 주지할 필요가 있다.

메타데이터 간의 상호운영성 확보를 목적으로 현재 가장 보편적으로 쓰이는 방식은 더블린코어를 기반으로 한 매핑이다. 분야별 이용자의 고급 정보서비스를 위해 상세 개발된 메타데이터 스키마를 기반으로 하되, 더블린코어 요소에 국한하여 매체간 혹은 분야간의 통합검색 서비스를 제공하는 것이 그 응용의 한 양태인데, 이럴 경우 공통 XML 스키마를 정하고 분산검색의 결과를 이 스키마에 근거하여 수집한 후 이용자에게 제공하는 방식을 취하게 된다. 그러나

만일 분야별 또는 매체별 상세 메타데이터 스키마의 구축을 생략하고 더블린코어를 기반으로 자원을 일괄 기술하는 경우라면 덤다운 원칙 적용에 따른 의미 충돌이 야기되지 않도록 스키마를 설계하는 것이 또한 중요하다.

다양한 메타데이터 간의 의미적 상호운영성을 강화하고 자원을 관리 또는 이용하는 다수 집단의 필요 충족을 극대화하기 위한 기술적 작업의 향후 선결과제는 시맨틱웹에 근거한 분산 메타데이터 사전의 구축이 되어야 할 것이다. 이러한 시맨틱웹의 발전에는 메타데이터 구문체계의 표준화와 온톨로지의 컨센서스라는 두 개의 중요한 전제가 따른다. 즉 우선적으로 모든 메타데이터 스키마와 애플리케이션 프로파일을 RDF 기반으로 정의하는 일, 그리고 DAML+OIL과 같은 온톨로지 언어가 시범적 운영단계를 넘어 더욱 정교해짐으로써 메타데이터 요소들 간의 연관관계를 명확히 규정하는 일이 그것이다.

또한 이러한 정의 단계를 거친 메타데이터 스키마와 애플리케이션 프로파일을 대상으로 한 MDR를 개발하여 메타데이터 스키마 설계자, 시스템 개발자 및 애플리케이션 프로그래머에게 제공될 필요가 있다. MDR 서비스는 중앙통제식 관리보다는 분야별 운영이 더 효율적일 것이나, 상호간의 연계 또한 병행되어야 할 중요한 과제로 보인다.

참고문헌

[1] Metadata Principles and Practicalities. Duval, E., et al. URL:http://www.dlib.org/dlib/april02/weibel/04weibl.html

[2] A Spectrum of Interoperability, Arms, W., et al., URL:http://www.dlib.org/dlib/january02/arms/01arms.html

[3] Truog, D. How the X Internet Will Communicate. The Forrester Report. December 2001.

[4] Keeping Dublin Core Simple:Cross-domain Discovery or Resource Description? Lagoze, C. URL:http://www.dlib.org/dlib/january01/01lagoze.html

[5] Guidelines for implementing Dublin Core in XML. Powell, A. URL: http://www.ukoln.ac.uk/metadata/dcmi/dc-xml-guidelines/

[6] An introduction to the Resource Description Framework, Miller, E. URL:http://www.dlib.org/dlib/may98/miller/05miller.html

[7] IETF RFC 1738 IETF(Internet Engineering Task Force). RFC 1738:Uniform Resource Locators(URL), ed. Berners-Lee, T., Masinter, L., & McCahill, M. 1994.

[8] Resource Description Framework(RDF) Schema Specification 1.0, URL:http://www.w3.org/TR/2000/CR-RDF-SCHEMA-20000327/

[9] The W3C XML Extensible Markup Language Woking Group Home Page, URL:www.w3.org/XML/

[10] Resource Description Framework(RDF) Schema Specification 1.0, URL:http:www.w3.org/TR/2000/CR-rdf-schema-20000327/

[11] Namespace Policy for the Dublin Core Metadata Initiative(DCMI), URL:http://dublincore.org/documents/dcmi-namespace/#DCQ

[12] R.Heery, M. Patel, "Application Profiles: mixing and maching metadata schemas", Aridne Issue 25, September 2000, Heery, R. & Patel, M. URL:http://www.ariadne.ac.uk/issue25/app-profiles/

[13] The SCHEMAS Project, Forum for Metadata Schema Implementers, URL:http://www.schemas-forum.org/

[14] An XML Schema Approach to Application Profiles, October 3 2000, Hunter, J., URL:http://archive.dstc.edu.au/maednad/appln_profiles.html

[15] EOR(Extensible Open Rdf)은 RDF 모델의 웹 검색 인터페이스 구축에 필요한 구성요소의 통합과 RDF에 의거한 검색 서비스 및 의미적 메타데이터 레지스트리의 구축에 필요한 기반의 제공을 그 목표로 하는 것이다.

[16] 현존하는 메타데이터 스키마와 애플리케이션의 검색에 필요한 종합적 MDR은 아직 구축되지 않았으나 몇 가지 시도되고 있는 예를 위해서는 부록을 참조하라

[17] A Metadata Registry for the Semantic Web. Heery R. & Wagner, H. URL:http://www.dlib.org/dlib/may02/wagner/05wagner.html

부 록

1. 메타데이터 레지스트리
- Dublin Core Metadata Registry
  (http://www.dublincore.org/registry/)
- UKLON Metadata Registry
  (http://www.uklon.ac.uk/registry)
- The Schemas Project
  (http://www.schemas-forum.org/registry)
- Dr. Sugimoto's Registry

2. RD기반 메타데이터 네임스페이스와 애플리케이션 프로파일
- The Schemas Project Metadata Set
  -NS : http://www.schemas-forum.org/registry/schemas/SCHEMAS/1.0/smes
  -AP : http://www.schemas-fourm.org/registry/schemas/SCHEMAS/1.0/smes-ap
- The RSLP Collection Level Description Metadata Set
  -NS : http://www.schemas-forum.org/registry/schemas/RSLP-CLD/1.0/cld
  -AP : http://www.schemas-fourm.org/registry/schemas/RSLP-CLD/1.0/cld-ap
- The BIBLINK CORE Metadata Set
  -NS : http://www.schemas-forum.org/registry/schemas/BIBLINK/1.0/bc
  -AP : http://www.schemas-fourm.org/registry/schemas/BIBLINK/1.0/bc-ap
- The DCMI Education Metadata Set
  -NS : http://dublincore.org/2000/08/22-dced
  -AP : http://www.schemas-fourm.org/registry/schemas/DCMI-Education/dced-ap
- MusicBrainz Metadata Initiative 2.0 Metadata Set
  -NS : http://musicbrainz.org/MM/

오 삼 균

1981 전북대학교 사범대학 영어교육학과 (학사)
1983 Villanova University(정보학 석사)
1995 Syracuse University(정보학 박사)
1989~1993 The King's College(전산학과 조교수)
1994~1998 University of Washington (정보대학원 조교수)
1998~2000 성균관대학교 문헌정보학과 (조교수)
2000~현재 성균관대학교 문헌정보학과 (부교수)
1994~2002 Syracuse University(하계강좌 초청교수)
2002 University of Texas at Austin (하계강좌 초청교수)
2002 University of Washington(하계강좌 초청교수)
관심분야:데이터 보델링, 메타데이터, 시멘틱웹, 디지털도서관
E-mail:samoh@skku.ac.kr