분산 디지털 도서관 시스템에서 XML을 이용한 가상문서의 표현과 처리[2)]

Representing and Processing Virtual Documents using XML in Distributed Digital Library Systems

강지훈, 맹성현, 이만호 충남대학교 컴퓨터과학과

Ji-Hoon Kang, Sung-hyun Myaeng, and Mann-Ho Lee
{jhkang, shmyaeng, mhlee}@cs.chungnam.ac.kr
Dept. of Computer Science, Chungnam National University

분산 환경에서 문서를 새로이 만들 경우, 인터넷 상의 여러 곳에서 문서들의 일부 또는 전부를 가지고 와서 사용하는 경우가 종종 있다. 또한 기업이 가지고 있는 분산 문서들을 서로 연관시켜 지식화하는 것은 글로벌시대의 기업 생존에 필수적인 사항이 되어가고 있다. 이러한 지식 생성기능을 분산 디지털 도서관 시스템에서 효율적으로 지원하기 위하여 제안된 새로운 문서 개념이 가상문서이다. 이 논문에서는 차세대 인터넷 언어의 표준으로 부각되고 있는 XML을 이용하여 가상문서를 표현하고, 그렇게 표현된 가상문서를 XML의 환경에서 처리하는 방법을 논한다. XML은 가상문서의 의미를 충분하고도 정확히 표현할 수 있고, 표준 처리 모델을 사용하여 용이하면서도 효율적으로 가상문서를 처리할 수 있으며, 또한 상용 브라우저와 같은 인터넷을 위한 여러 도구를 사용함으로써 인터넷을 통하여 가상문서에 접근할 수 있도록 하는 등의 다양한 장점을 가지고 있다.
중심어 : 가상문서, 디지털도서관, 인터넷, XML, 지식

1.서론

인터넷의 급격한 발달과 더불어 문서의 배포와 공유의 속도는 날로 더 가속화 되어가고 있으며, 그 범위 역시 개인 가정에까지 이르고 있다. 이러한 분산 환경에서 새로운 문서를 만들 경우, 인터넷 상의 여러 곳에서 문서들의 일부 또는 전부를 가지고 와서 사용하는 경우가 종종 있다.

예를 들어 학생들은 교사가 요구하는 보고서를 작성하기 위하여, 인터넷의 여러 곳에 흩어져 있는 문서들을 검색하며, 그중에서 원하는 전문을 받아서 필요한 부분을 잘라내어 자기의 보고서의 일부로 인용하는 것과 같은 행위를 할 수 있다. 또한 기업은 분산된 문서들을 서로 연관시켜 지식화하는 것은 글로벌 시대의 기업 생존에 필수적인 사항이 되어가고 있다. 각 부서에서 발생하는 문서 형태의 지식은 서로공유가 되어야 하며, 공유를 원활히 하고 못하는 것이 기업의 경쟁력에 직결되는 것이 현실이다.

2) 이 연구는 충남대학교 소프트웨어연구센터의 재정지원을 받았음

그러므로 디지털 도서관은 디지털 문서를 검색하고 전문을 제공하는 것과 같은 본연의 기능 이상의 것을 수행하기를 요구받는다. 즉, 문서로 되어 있는 지식을 쉽게 공유하고 생성할 수 있는 방법을 제공할 필요가 있다. 이러한 지식 생성 기능을 분산 디지털 도서관 시스템에서 효율적으로 지원하기 위하여 제안된 새로운 문서 개념이 가상문서이다[2]. 가상문서는 링크로만 이루어지는 문서로서, 각 링크는 분산 환경에 흩어져 존재하는 문서의 일부 또는 전부를 가리킨다. 이렇게 함으로써 지식의 공유를 쉽게 표현할 수 있으며, 실제 문서를 링크로만 가지고 있음으로써 저장 공간이나 통신량을 감소시키는 효과를 얻을 수 있다.

이 논문에서는, 가상문서를 기반으로 하여 지식 공유를 효율적으로 하는 것을 목적으로 하는 디지털 도서관에서 XML[6]로 가상문서를 표현하고, 그렇게 표현된 가상문서를 XML의 환경에서 처리하는 방법을 논한다. 가상 문서의 개념을 나타내기 위하여 사용되는 표현 언어는 표현력, 처리효율성, 및 접근성이 우수하여야 한다. 우리는 요구 조건에 맞는 언어로서 XML을 택하였다. XML은 문서나 정보의 구조를 나타내는 표현력이 우수하며, DOM[10]과 같은 표준 처리 모델을 사용함으로써 용이하면서도 효율적으로 가상문서를 처리할 수 있다. 또한, XML은 인터넷 문서 언어로서의 HTML이 담당하지 못하는 영역을 위한 차세대 인터넷 문서 언어의 표준으로 부각되고 있다는 점에서 인터넷에서의 활용성이 매우 높을 것으로 예측된다. 따라서 상용 브라우저와 같은 다양한 웹 도구를 사용함으로써 인터넷을 통하여 가상문서에 쉽게 접근할 수 있다. 가상문서를 XML로 표현함으로써 얻게되는 이러한 장점은 우리의 지식 공유를 위한 디지털 도서관의 기능을 한 차원 높일 것으로 기대한다.

다가문서 (multivalent document) 모델[4]과 FEDORA [3]에서의 디지털 객체(Digital Object) 개념은 디지털 도서관을 위한 문서 구조를 정의한다는 점에서 우리의 연구와 공통점이 있다. 전자의 경우, 문서는 다양한 형태의 층으로 이루어지며, 각 층은 "behavior"라 부르는 프로그램의 지원을 받는다. 후자의 경우에는, 디지털객체는 사용자 측면에서는 내부 구조가 감추어져 캡슐화 되어 있으며 "disseminator"라 부르는 서비스 인터페이스의 집합을 통하여 사용자가 디지털 객체에 개념적으로 쉽게 접근할 수 있도록 하고 있다.

우리의 가상문서 모델은 "behavior"나 "disseminator"라는 처리 개념이 없는 좀 더 단순한 형태를 취하고 있다. 반면, 다양한 링크를 지원함으로써, 기존의 문서를 재사용 하여 새로운 문서를 쉽게 만들 수 있다. 한편, 기존의 두 접근 방식은 자료를 틀 안에 넣고 구조화하는 데에 주안점을 둔 데 비하여, 우리의 방식은 자료를 연결하고 확장하는 데에 초점을 맞추었다.

논문의 구조는 다음과 같다. 2 절에서는 가상문서의 개념을 설명한다. 3 절에서는 가상문서를 XML로 표현하는 이유와 방법을 설명한다. 4 절에서는 가상문서를 디지털도서관 환경에서 처리하는 방법을 알아본 후에, 마지막으로 5 절에서 결론을 맺는다.

2.가상문서

2.1 가상문서의 개념 및 정의

가상문서 개념은 인터넷 상에 존재하는 문서 전체 또는 그 일부를 재사용하여 새로운 문서를 쉽게 만들 수 있도록 하려는 것이다. 이를 달성하기 위하여 링크를 이용하여 분산 환경에 존재하는 문서들을 모아놓는 방법을 사용한다.

가상문서는 **허브**(hub)와 **스타일쉬트**(stylesheet)로 구성된다. 허브는 문서의 구조와 내용을 가지고 있으며, 스타일은 가상문서 전체 및 그 구성요소를 위한 스타일이다. 허브는 내포링크 순서 (a sequence of embedding links), 참조링크집합 (a set of referential links), 그리고 메타데이타 (metadata)의 세 개로 구성된다.

내포링크 순서는 하나 이상의 내포링크의 순서로 이루어진다. 내포링크는 그 링크가 가리키는 문서가 불러들여져서 그 순서대로 하나의 문서를 이루게 됨을 의미한다. 인터넷 상에 존재하는 문서는 모두 내포될 수 있으며, 특별히 문서의 부분만을 내포할 수 있는 것이 특징이다.

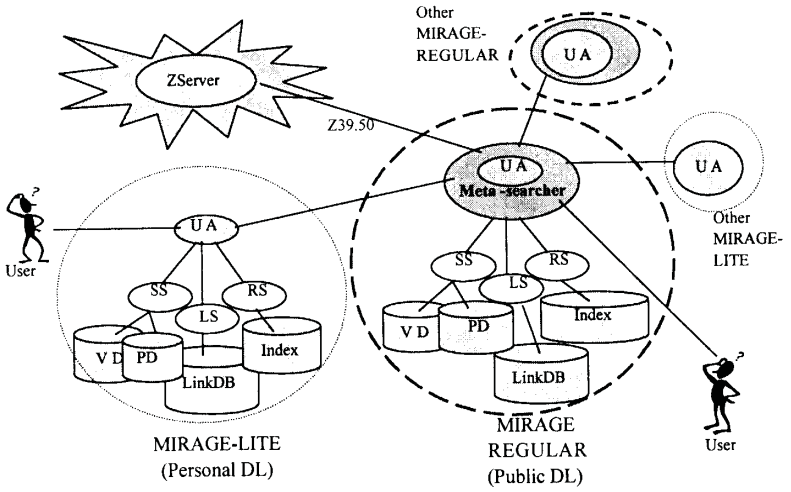
참조링크 집합은 여러 개의 참조링크를 모아놓은 집합이다. 개념적으로는 HTML문서에서 사용하는 링크의 의미를 확장한 것이다. 링크가 출발 앵커 (source anchor)와 함께 존재하지 아니하고 별도로 존재하며, 따라서 CR-ROM과 같은 읽기 전용 문서일지라도 참조 링크를 그 위에 설정할 수 있다. 다중링크를 지원하며, 총칭 링크(generic link) 개념을 지원한다.

메타데이타는 가상문서에 관한 메타데이타를 Dublin Core[1]의 15 개 항목으로 기술하기 위한 것이다.

이러한 링크로 이루어진 가상문서를 사용함으로써 다음과 같은 것을 기대할 수 있다. 즉, 새롭게 구성되는 가상문서는 모든 구성 요소의 복사본을 가지고 있을 필요가 없으므로, 내포해야 할 문서가 원격지에 있더라도 저장공간을 확보하지 않아도 되며, 가상문서를 네트워크에서 전송할 때에 통신 시간상의 득을 볼 수 있다. CD-ROM과 같은 문서의 읽기 전용 속성을 변경시키지 않고도 링크를 정의할 수 있다. 그리고, 모아진 문서들 전체에 대하여 메타 정보를 부여하는 것이 가능하다.

2.2 가상문서를 지원하는 디지털 도서관

우리는 가상할 수 있도록 문서의 목적을 충분히 달성하기 위하여, 가상문서를 보다 효율적으로 처리할 수 있는 [그림 1]과 같은 MIRAGE 디지털도서관 시스템을 설계하였다 [2].



[그림 1] MIRAGE 디지털도서관 환경.

MIRAGE에는 두 가지 버전의 디지털도서관 시스템이 있는데, 정규 버전으로서 완전한 기능을 갖춘 MIRAGE-Regular와 개인용 버전인 MIRAGE-Lite이다. 두 버전의 기본 기능은 같으나 Lite는 개인용도서관의 구축에 초점을 맞추었다. 즉, Regular의 기능 중 메타 검색과 같은 일부기능을 없애고 모든 시스템 구성요소를 단일 시스템 안에 설치되도록 함으로써 PC와 같은 저가형 개인용 시스템에 구현이 되도록 설계하였다. Regular에 비해 부족한 기능은 Regular에 접속하여 도움을 받도록 하였다. 두 버전 모두 다음의 구성요소로 이루어져 있다. 사용자 에이전트(UA), 검색 서버 (RS), 저장서버 (SS), 및 링크 서버 (LS)이다.
UA는 핵심적인 역할을 하는 구성요소로서 사용자로부터 질의를 받아 그 결과를 사용자에게 돌려준다. 질의는 RS에게 전달하여 검색을 요청을 한다. RS는 검색을 하여 결과를 얻는다. SS는 문서의 내용을 전달하며, LS는 링크 정보를 전달한다.
가상문서의 관점에 보면 가상문서 저작도구를 이용하여 만들어진 문서는 UA로 전달이 되며, UA는 문서에서 정보를 추출하여 링크정보는 LS로 색인정보는 RS로 보내며, 문서 자체는 SS를 통하여 저장한다. 검색 결과로서 사용자가 가상문서를 요청하였을 경우에는, 그 요청을 받은 UA는 SS를 통하여 가상문서의 내용을 전달받으면 사용자 브라우저가 인식할 수 있는 형태로 변환을 하여 사용자에게 전송한다.

3. XML을 이용한 가상문서의 표현

3.1 XML의 개요

SGML[5]은 우수한 표현력에도 불구하고, 구문의 복잡성으로 인하여 파서와 같은 SGML을 지원하는 도구 소프트웨어의 크기가 커서 인터넷 환경에는 적합하지 않다. 한편, 인터넷 문서의 마크업 언어 표준으로 자리잡은 HTML은 SGML의 응용 예, 즉 하나의 DTD (Document Type Declaration)로 정의됨으로써 미리 정해놓은 태그들만 사용해야 하며, 따라서 다양한 문서 구조를 나타내기에는 한계를 드러내었다. 또한 스타일 정보가 문서 내용 정보와 혼재하여 있음으로 인하여 문서의 스타일을 하나의 형태로 국한시켜서 필요에 따라 문서의 스타일을 쉽게 바꿀 수 있도록 하는 융통성을 제한하였다.

이러한 SGML과 HTML의 단점을 극복하기 위한 대안으로, W3C (WWW Consortium)는 1998년 2월에 XML[6] (eXtended Markup Language)을 차세대 인터넷 문서를 위한 표준 마크업 언어로 권고하였다.

XML은 SGML의 부분집합으로 정의되었다. DTD를 이용하여 문서의 구조를 정의하고 그에 맞게 문서를 작성하는 SGML의 근본 정신은 그대로 두면서, 내용 모델에서 'and' 연산자를 허락하지 않으며, 태그를 생략할 수 없도록 하는 등 일부 사용하기 복잡한 문법을 삭제하거나 수정하여 인터넷 환경에서 쉽게 사용되고 처리될 수 있도록 하였다. 그러므로 XML의 표현력은 SGML에 비하여 약간 떨어지나 HTML보다는 월등하며, SGML보다 간결해진 구문에 의하여 처리의 효율성이 제고되었으며, 내용과 스타일 정보의 구분으로 인하여 HTML보다 문서 내용의 재활용성이 우수하여졌다. XML은 이제 인터넷 기반 하이퍼텍스트 문서뿐만이 아니라 디지털도서관, EDI (Electronic Data Interchange), 전자상거래, CALS (Commerce at Light Speed) 등 인터넷에서 정보 교환이 일어나는 곳이면 어디든지 활용될 수 있다.

XML은 단순히 문서의 구문만을 정의한다. 그러므로 보다 풍부한 의미를 나타낼 수 있도록 링크나 스타일쉬트 등 여러 가지 관련 표준들의 제정이 함께 추진되고 있으며, 일부는 이미 표준으로 확정이 되었다.

링크를 위한 표준으로는 XLink[7], XPointer[8], 및 XPath[9]가 있다. XLink를 이용하여 XML 문서에서 양방향(bi-directional) 링크, 다중(multi-directional)링크, 내포(embedding) 링크, out-of-line링크 등 HTML에서는 불가능한 링크들을 만들 수 있다. XPointer나 XPath를 사용하여 ID가 선언되지 않은 원소를 참조할 수 있으며, 더 나아가 문서의 원소가 아닌 곳, 즉 원소 내의 내용의 일부와 같은 곳도 참조가 가능하다. 이러한 참조 가능한 모든 곳에 XML 링크의 앵커를 설정할 수 있다.

Namespaces[11]는 다양한 응용을 수용함으로써 발생하는 이름의 유일성 문제를 해결하기 위하여 만들어졌으며, DOM[10]은 XML 문서의 표준처리모델 및 인터페이스를 정의하였다.

스타일 정보를 위하여 XSL[12]과 XSLT[13]가 있다. XSLT는 XML 문서의 변환을 정의하는 언어이다. 스타일을 적용하려는 문서의 형태를 먼저 변환할 필요가 있을 경우 사용되며, XSLT 처리기는 XSL처리기의 전처리기 역할을 한다. XSL은 XML 문서에 대한 스타일쉬트를 표현하는 언어이다. 패턴을 이용하여 스타일을 적용할 원소들을 정의하며, 규칙을 이용하여 실제 스타일 정보를 명시한다.

DTD는 문서의 구조는 잘 정의할 수 있으나 그 내용의 형(type)을 정의하기에는 그 기능이 매우 약하다. 예를 들어 나이를 나타내는 원소라면 그 원소의 내용으로는 0보다 큰 정수가 들어가야 하나, DTD로는 이를 정의할 수 있는 방법이 없다. 특별히 EDI나 전자상거래와 같은 경우 유효한 정보만을 허락하기 위해서 원소에 따라 그 내용의 형과 유효한 값의 범위 등의 조건을 정의하는 것은 매우 필요한 기능이다. 이를 위하여 XML Schema가 만들어졌다. XML Schema를 이용하여 DTD와 같이 문서의 구조도 정의하며, 더불어 원소의 내용에 관한 형을 정의할 수 있다.
그밖에도, 메타데이타를 위한 RDF, 멀티미디어 객체의 동기화를 위한 SMIL, 수식을 위한 MathML, 그래픽 객체의 벡터정보를 나타내기 위한 VML, XML 문서에 대한 질의 언어인 XML-QL 등 다양한 XML 관련 표준이 있다.

3.2 가상문서를 표현하는 언어로서의 XML

가상문서를 지원하는 디지털도서관 시스템은 사용자에게 가상문서의 장점을 충분히 제공함과 동시에 문서를 효율적으로 처리해야 한다. 그러므로 그 내부 표현 방법은 중요한 관건이며, 이를 위하여 다음과 같은 사항을 고려할 필요가 있다.

● **표현력 :** 가상문서 개념의 모든 기능이 표현될 수 있어야 한다. 문서의 의미와 구조가 정확히 표현되어야 하며 모호성이 없이 이해될 수 있어야 한다.

●**처리효율성 :** 가상문서를 지원하는 시스템은 가상문서를 효율적으로 처리할 수 있어야 한다. 쉽게 구문 분석하고, 실행시간에 내부 자료구조를 만들고, 처리하는 등의 작업이 효과적으로 이루어질 수 있어야 한다.

●**접근성 :** 가상문서는 인터넷 상의 다양한 도구를 이용하여 접근이 가능하여야 한다. 예를 들어 넷스케이프 네비게이터나 인터넷 익스플로러와 같은 상용 브라우저를 사용하여 화면에 보여주는 것과 같은 일이 가능하여야 한다.

처음 두 조건만을 고려한다면 다양한 선택의 방법이 있을 것이다. 비록 세 번째 조건은 그 선택의 폭을 제한하고는 있지만, 그동안 인터넷에서 HTML이 끼친 영향력과 앞으로의 인터넷의 발전을 미루어볼 때에 이 조건은 매우 중요하다고 할 수 있다.
먼저, HTML은 우리의 목적에는 부적합한 언어라고 판단하였다. 인터넷 문서의 표준 역할을 하고는 있으나, 제한된 태그만을 사용한다는 점 등, 가상문서의 개념을 나타낼 수 있을 만큼 충분한 기능을 가지고 있지를 못하다.
고유의 언어 형태를 만들어서 사용할 수 도 있으나 두 번째와 세 번째 조건을 만족시키지 못하므로 이 또한 적합하시 않은 것으로 판단하였다. 고유의 표현 언어로 작성된 문서를 처리하기 위해서는 모든 필요한 도구를 개발하거나, 또는 인터넷 표준 브라우저 등과 같은 표준 도구들을 이용하여 처리할 수 있도록 추가적인 변환작업이 필요할 것이다.
우리는 세 가지 조건을 모두 만족하면서 가상문서를 잘 표현할 수 있는 언어로서 XML을 선택하였다. 다음은 XML이 각 조건을 어떻게 충족시키는지 설명하고 있다.

● 앞에서 언급하였듯이 XML은 표현력이 SGML보다는 약간 떨어질 뿐, HTML보다는 월등하다. 그러므로 가상문서를 나타내기에는 충분하다.
● XML은 SGML의 문법보다 단순하여 XML 문서를 인터넷에서 효율적으로 처리할 수 있는 장점이 있다. XML을 위한 파서가 공개 소프트웨어로 이미 많이 나와 있다. 구문 분석의 결과는 XML 문서의 메모리 모델인 DOM 트리로 표현이 되며, 표준 DOM 인터페이스를 통하여 문서를 처리할 수 있다.
● 인터넷에서 표준 도구를 이용하여 XML문서로 접근이 용이하게 위해서는 XML이 HTML처럼 인터넷의 표준으로 자라 잡아 많은 XML에 대한 W3C의 중요한 목표 중의 하나이다. 비록 아직까지 브라우저와 같은 XML을 지원하는 도구들의 개발이 매우 느리기는 하지만, 우리가 알기로는, 많은 사람이 XML은 성공할 것이라고 믿고 있다

3.3 가상문서를 위한 XML DTD

가상문서를 XML로 표현하려면 가상문서의 구조를 정의하는 DTD가 필요하다. 가상문서는 허브와 스타일쉬트로 구성된다. 그러므로 각각의 DTD를 정의한다. DTD의 내용을 명확하게 표현하기 위하여 arial 문자체를 사용하여 나타내었다.

3.2.1 허브를 위한 DTD

[그림2]는 허브를 위한 DTD를 보여준다. 루트 원소는 VDocHub이며, 내포 링크 순서를 위한 ELinkSeq, 참조링크 집합을 위한 RLinkSet, 및 Dublin Core 메타데이타를 위한 Metadata의 세 개의 원소로 구성된다.

<!ELEMENT VDocHub
(RLinkSet, ELinkSeq, Metadata) >

ELinkSeq는 하나 이상의 ELink들의 순서로 이루어진다. ELink가 하나도 없으면 가상문서는 내용이 하나도 없게 되므로 의미가 없다. ELink들이 나열되는 순서가 바로 가상문서를 보여줄 때 내포 문서들이 나열되는 순서가 된다.

<!ELEMENT ELinkSeq (ELink +)>

ELink는 속성 href의 값으로 내포링크의 목적지를 정의하게 된다. 출발지는 ELink 자신이다. 내포 링크의 목적지는 문서 전체가 될 수도 있고, 그 문서의 일부분만을 가리킬 수도 있다. 텍스트나 이미지의 일부분만 내포하는 것이 가능하다, 다른 속성으로 title과 role이 있으며, 이들은 링크의 의미적 정보를 가지게 된다. owner와 date 속성은 링크를 만든 사람과 만들어진 날짜를 위한 것이다. category 속성은 링크의 목적지에 대한 내용을 분류하기 위한목적으로 사용한다. 속성 actuatedefault는 가상문서가 사용자에게 보여질 때, ELink로 정의된 내포된 문서가 자동으로 내포될 것인지 아니면 사용자가 요청할 때에 내포할 것인지를 나타낸다. 속성 autoDelete는 내포되는 문서가 없어져서 링크가 더 이상 의미없는 경우에 디지털 도서관 시스템이 자동적으로 링크를 삭제하는 것을 허락할 것인지를 표시한다.

```
<?xml version='1.0' encoding="euc-kr"?>
<!-- DTD for VDoc Hub. 1999-10-28 -->

<!ELEMENT VDocHub    (ELinkSeq, RLinkSet, Metadata) >

<!-- Embedding Links   -->
<!ELEMENT ELinkSeq   (ELink)+ >

<!ELEMENT ELink                      EMPTY >
<!ATTLIST ELink
          href          CDATA        #REQUIRED
          role          CDATA        #IMPLIED
          title         CDATA        #IMPLIED
          owner         CDATA        #IMPLIED
          date          CDATA        #IMPLIED
          category      CDATA        #IMPLIED
          actuatedefault             (user|auto) "user"
          atuoDelete    (NO|YES)     "NO" >

<!-- Reference Links -->
<!ELEMENT RlinkSet      (RLink*) >
<!ELEMENT RLink         (Source, Destination+) >
<!ATTLIST RLink
          role          CDATA        #IMPLIED
          title         CDATA        #IMPLIED
          owner         CDATA        #IMPLIED
          date          CDATA        #IMPLIED
          category      CDATA        #IMPLIED
          showdefault     (new|parsed|replace)   "replace"
          actuatedefault  (user|auto)            "user" >

<!ELEMENT Source        EMPTY >
<!ATTLIST Source
          Is_generic    (NO|YES)     "NO"
          href          CDATA        #REQUIRED
```

```
          role          CDATA        #IMPLIED
          title         CDATA        #IMPLIED
          autoDelete    (NO|YES)     "NO" >

<!ELEMENT  Destination  EMPTY >
<!ATTLIST  Destination
           href         CDATA        #REQUIRED
           role         CDATA        #IMPLIED
           title        CDATA        #IMPLIED
           autoDelete   (NO|YES)     "NO" >

<!--  Meta-data / Dublin Core type -->
<!ELEMENT  Metadata
           (DC_TITLE?, DC_CREATOR?, DC_SUBJECT?,
           DC_DESCRIPTION?, DC_PUBLISHER?,
           DC_CONTRIBUTOR?, DC_TYPE?, DC_DATE?,
           DC_FROMAT?, DC_IDENTIFIER?, DC_SOURCE?,
           DC_LAGUAGE?, DC_RELATION?,
           DC_COVERAGE, DC_RIGHTS?) >
<!ELEMENT  DC_TITLE   EMPTY >
<!ATTLIST  DC_TITLE
           value        CDATA        #IMPLIED >
<!ELEMENT  DC_CREATOR                EMPTY >
<!ATTLIST  DC_CREATOR
           value        CDATA        #IMPLIED >
<!ELEMENT  DC_SUBJECT                EMPTY >
<!ATTLIST  DC_SUBJECT
           value        CDATA        #IMPLIED >
<!ELEMENT  DC_SUBJECT                EMPTY >

           . . . . . .

<!ELEMENT  DC_RIGHTS EMPTY >
<!ATTLIST  DC_RIGHTS
           value        CDATA        #IMPLIED >
<!-- ------------------------------------------ -->
```

[그림 2] 가상문서의 허브를 위한 DTD.

RLinkSet은 다음과 같이 0 개 이상의 RLink로 구성된다.

<!ELEMENT RLinkSet (RLink*)>

가상문서에 새롭게 추가될 참조링크는 모두 여기에 RLink로 정의된다. 가상문서는 문서 내용을 직접 가지고 있지 않으므로 여기서 정의되는 모든 참조 링크는out-of-line 링크이다. 예를 들어, 내포되는 문서가 읽기 전용이지만 그 위에 참조 링크를 설정하려면, 이 새로운 링크는out-of-line 링크로 정의될 수밖에 없으며, 그러므로 RLink로 정의될 수 있다. 문서 안에 있는 'W3C' 라는 모든 단어에 대하여 같은 목적지인 W3C 홈페이지로의 참조 링크를 설정하기 위한 총칭 링크의 경우 역시 out-of-line의 성격을 가질 수밖에 없다.

RLink는 한 개의 Source 앵커와 하나 이상의 Destination 앵커로 구성된다.

<!ELEMENT RLink (Source, Destination+)>

Source는 총칭 링크인지 여부를 나타내기 위한 속성으로 is_generic을 가지고 있다. Source가 총칭형이면 is_generic은 'YES'이며, Source의 위치를 나타내는 속성 href의 값은 총칭 표현식이 된다. 속성 role과 title은 링크의 의미적 정보를 가지게 된다, 속성 showdefault는 사용자가 링크를 택하였을 경우 목적지의 문서를 가져온 후에 사용자에게 어떻게 보여줄 것인지를 정의한다. 그 값으로 'new'는 새로운 창을 띄워 그곳에 보여달라는 것이며, 'parsed'는 현 위치에 내포시키라는 것이며, 'replace'는 현재의 창의 내용을 지우고 그곳에 보여달라는 의미이다.

VDocHub의 세 요소 중 마지막 구성요소인 Metadata는 가상문서의 메타데이타를 기술하기 위하여 더블린 코어의 15개 항목을 정의한다.

3.2.1 스타일쉬트를 위한 DTD

[그림 3]은 스타일쉬트를 위한 DTD를 보여준다. VDocStyle은 하나 이상의 ELinkStyle로 구성된다. ElinkStyle의 개수는 관련 가상문서의 ELinkSeq에 있는 ELink의 개수와 일치하는 것을 정상으로 본다.

<!ELEMENT VDocStyle (ELinkStyle+)>
ELinkStyle은 해당 ELink에 의하여 내포되는 문서 블록에 대한 스타일 정보 세 가지를 정의한다.
<!ELEMENT ELinkStyle (align?|font?|color?)>
align은 문단의 정렬방법, font는 글자체, 그리고 color는 색상을 정의한다.

```
<?xml version='1.0' encoding="euc-kr"?>
<!--  DTD for VDoc Stylesheet. 1999-10-28 -->

<!ELEMENT  VDocStyle  (ELinkStyle+) >

<!ELEMENT  ELinkStyle  (align?|font?|color?) >

<!ELEMENT  align       EMPTY >
<!ATTLIST  align
           value       (left|center|right)    'left' >

<!ELEMENT  font        EMPTY >
<!ATTLIST  font
           name        CDATA          #IMPLIED
           size        CDATA          #IMPLIED
           bold        (true|false)   'false'
           italic      (true|false)   'false' >

<!ELEMENT  color       EMPTY >
<!ATTLIST  color
           value       CDATA          #REQUIRED >
```

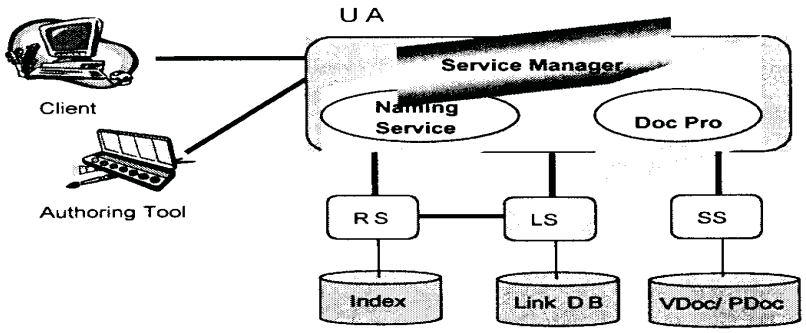
[그림 3] 가상문서 스타일쉬트를 위한 DTD

4. 가상문서의 처리

4.1 User Agent의 구조

가상문서를 처리하기 위한 User Agent(UA)의 구성은 [그림 4]와 같다. 가상문서 저작도구에 의하여 생성된 가상문서는 UA의 Service Manager (SM)에게 전달된다. SM은 이것을 Document Processor(DocPro)에 보내며, DocPro는 XML 파서를 사용하여 구문분석을 하고 DOM 트리를 만든다. Naming Service를 사용하여 가상문서와 그 내포문서에 대한 블록 ID를 배정하며, 링크 정보는 추출하여 LS로 보내며, 색인 정보는 RS로 보낸다. 또한 가상문서를 SS로 보내어 디스크에 저장한다.

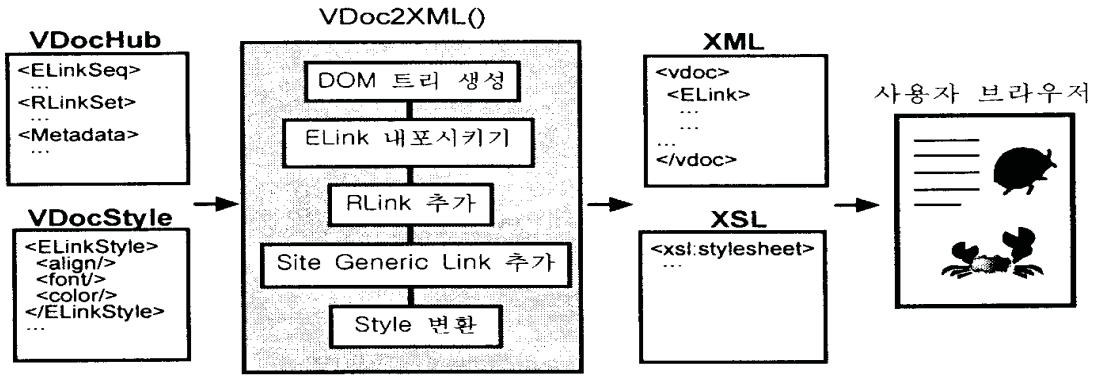
검색화면을 통하여 작성된 사용자의 검색질의는 UA에게 보내지면, UA의 SM을 통하여 RS로 전달된다. RS는 색인정보와 LS의 도움을 받아 링크기반 검색을 하며, 검색 결과는 SM을 통하여 사용자에게 돌려준다. 사용자는 검색 결과로 가상문서의 전문을 보기를 선택하면, 이 요청은 SM으로 전달이 되며, SM은 DocPro에게 전달한다. DocPro는 가상문서의 위치를 파악하여 자기의 SS에 없을 경우에는 SM에게 요청하여 해당 사이트와 접속하여 가져오도록 하며, 아니면 SS에게 요청하여 지역 디스크에서 읽어서 가져오도록 한다. DocPro가 가상문서를 받게되면, 처리하여 사용자가 화면에 볼 수 있는 XML 문서로 만든다. 이때 내포 링크가 문서의 부분만을 가리킬 경우에는, SS에게 요청하여 부분을 얻어내는 작업을 한다. 만들어진 문서를 SM을 거쳐 사용자에게 보내면 사용자는 XML브라우저를 이용하여 가상문서를 볼 수 있다.



[그림 4] 가상문서 처리를 위한 UA의 구성.

4.2 가상문서의 변환

DocPro의 기능 중에서 가장 핵심적인 역할을 하는, 가상문서를 사용자가 볼 수 있도록 변환하는 처리기의 구성은 [그림 5]와 같다. 처리기에는 가상문서의 허브와 스타일이 주어진다. 이들은 XML 파서를 사용하여 DOM 트리로 만든다. 이 DOM트리로부터 변환에 필요한 정보를 추출한다. 추출된 정보에서 내포 링크를 순차적으로 처리하여 문서를 가져와서 결합시켜 하나의 문서를 만든다. 여기에 참조 링크와 총칭 링크를 추가한다. 한편 스타일 정보를 이용하여 변환된 문서를 위한 XSL스타일쉬트를 만든다. 스타일쉬트는 SS로 보내어 저장하고, 변환된 XML 문서에 XSL의 위치정보를 추가하여 변환을 완료한다.



[그림 5] 가상문서를 사용자가 볼 수 있는 XML 문서로 변환하기.

4.2 링크 데이타베이스

우리의 디지털 도서관 시스템은 링크 데이타베이스를 관리한다. 이는 가상문서에 포함된 링크를 이용하는 링크 기반 검색과 같은 일을 효율적으로 처리 할 수 있게 해준다. 링크 데이타베이스는 가상문서에 있는 모든 링크를 유지 관리한다. 가상문서의 링크에 변경이 생기면 링크 서버 (LS)는 문서 처리기 (DocPro)로부터 변경 정보를 넘겨받아 링크 데이타베이스에 반영한다.

가상문서의 개념 중에서 중요한 기능 중의 하나가 사이트 총칭 링크이다. 그러나 만일 사이트 총칭 링크를 모든 가상문서가 가지고 있게 되면 변경이 일어날 경우, 불일치성의 문제가 발생한다.

이 문제는 사이트 총칭 링크를 링크 데이타베이스 한곳에 모음으로써 해결할 수 있다. 사이트총칭 링크를 관리하는 도구가 필요하게 되는데 이는 가상문서 저작 도구가 지원할 수 있다.

5.결론

가상문서는 사용자가 인터넷에 산재되어 있는 기존의 문서를 이용하여 손쉽게 새로운 문서를 만들 수 있는 방법론을 제공한다. 우리는 이 개념을 XML로 표현함으로써 표현력, 처리 효율성, 및 인터넷에서의 접근성을 확보하였다. 가상문서를 나타내기 위한 DTD를 정의하였으며, 가상문서 개념을 지원하는 MIRAGE 디지털도서관 시스템을 설계하였다. 현재 CORBA와 Java를 이용하여 프로토타입을 구현하는 중이다.

6.참고문헌

[1] T. Baker, "Language for Dublin Core," D-lib Magazine, Dec. 1998.
[2] S. H. Myaeng, M.-H. Lee, and J.-H. Kang, "Virtual Documents: a New Architecture for Knowledge Management in Digital Libraries," Proc. Asian Digital Libraries Conf., Taipei, Taiwan, Nov. 1999.
[3] S. Payette and C. Lagoze, "Flexible and Extensible Digital Object and Repository Architecture (FEDORA)," Proc. the 2nd European Conf. on Digital Libraries, Heraklion, Greece, Sept. 1998.
[4] T. Phelps and R. Wilensky, "Towards Active, Extensible, Networked Documents: Multivalent Architecture and Applications," Proc. 1st ACM Int'l Conf. on Digital Libraries, Bethesda, MD, Mar. 1996.
[5] ISO 8879, Information Processing - Text and Office Systems - Standard General Markup Language (SGML), 1986.
[6] W3C, Extensible Markup Language (XML) 1.0, Recommendation, Feb. 1998. (www.w3.org/TR/REC-xml)
[7] W3C, XML Linking Language (XLink), Working Draft, July 1999. (www.w3.org/TR/xlink)
[8] W3C, XML Pointer Language (XPointer), Working Draft, July 1999. (www.w3.org/TR/xpointer)
[9] W3C, XML Path Language (XPath), Proposed Recommendation, Oct. 1999. (www.w3.org/TR/xpath)
[10] W3C, Document Object Model (DOM) Level 1, Recommendation, Oct. 1998. (www.w3.org/TR/REC-DOMLevel-1)
[11] W3C, Namespaces in XML, Recommendation, Jan. 1999. (www.w3.org/TR/REC-xml-names)
[12] W3C, Extensible Stylesheet Language (XSL) Specification, Working Draft, Apr. 1999. (www.w3.org./TR/WD-xsl)

[13] W3C, XSL Transformations (XSLT), Recommendation, Oct. 1999. (www.w3.org./TR/xslt)