클라이언트 상의 Well-Formed XML 문서 처리 시스템의 설계 및 구현

송 종 철† · 문 병 주† · 홍 기 채† · 정 현 수† · 김 규 태†† · 이 수 연†††

요 약

XML은 SGML의 기능 및 구문을 인터넷상에서 쉽게 실용적으로 사용하기 위하여 단순화시킨 메타언어이며 XSL, XLL 및 Xpointer등과 함께 사용되고 있다. 또한 DTD를 포함하지 않는 Well-Formed XML을 제공하여 XML문서를 웹상에서 간편하게 사용할 수 있다. 그러나 브라우즈 기능과 XLL의 확장 링크 기능, DTD 생성 기능을 통합하여 Well-Formed XML 문서를 효율적으로 처리할 수 있는 시스템이 제공되지 않았다. 본 논문에서는 클라이언트에서 DTD를 포함하지 않는 Well-Formed XML 문서를 효율적으로 처리할 수 있도록 Well-Formed XML 뷰어와 자동 DTD 생성기, Non-Validating 파서, XLL 처리기 등으로 구성된 시스템을 설계 및 구현하였으며, 확장 기능을 가지는 XLL과 Xpointer 처리, XLL의 링크 항해 시 동일 클래스의 Well-Formed XML문서들에서 DTD를 추출하여 자동으로 생성하는 기능에 초점을 맞추었다. 링크 처리 시, 링크 주소 지정 방식은 ID와 Xpointer에 의한 직접 주소 지정 방식을 사용하였다. 본 시스템의 구현 결과, XLL 기능의 유효성을 확인하였고 같은 루트 엘리먼트를 갖는 동일한 클래스의 Well-Formed XML문서들로부터 DTD를 추출하여 일반화된 DTD를 생성하였다.

The Design and Implementation of the System for Processing Well-Formed XML Document on the Client-side

Jong-Chul Song† · Byung-Joo Moon† · Gi-Chai Hong† · Hyun-Soo Cheong† · Gyu-Tae Kim†† · Soo-Youn Lee†††

ABSTRACT

XML is a meta-language as SGML and also can be constructed as an Internet version of simplified SGML, being used in conjunction with XLL, Xpointer and XSL. Also W3C established DTDless Well-Formed XML document to use XML document on the Web. But it isn't offered system that consists of browsing, link and DTD generating facility, and efficiently processes DTDless Well-Formed XML document. This paper studies on an implementation and design of system to process DTDless Well-Formed XML document on the client-side. This system consists of Well-Formed XML viewer displaying Well-Formed XML document, XLL Processor processing XLL and Auto DTD generator constructing automatically DTDs based on multiple documents of the same class. This study focuses on automatic DTD generation during hyperlink navigation and an implementation of extended links based on XLL and Xpointer. ID and Xpointer location address are used as the address mode in the links. As a result of implement of this system, it conforms to validation of extended link facilities, extracts DTD from Well-Formed XML Documents including same root element at the same class and constructs generalized DTD.

※ 본 논문은 1999학년도 광운대학교 교내학술비 지원을 받아 수행된 연구결과입니다.

† 정 회 원 : 한국전자통신연구원 컴퓨터통신연구팀
†† 정 회 원 : 광운대학교 대학원 컴퓨터공학과
††† 정 회 원 : 광운대학교 컴퓨터공학과 교수

논문접수 2000년 6월 8일, 심사완료 : 2000년 9월 28일

1. 서 론

최근 인터넷의 발전에 따른 WWW(World Wide Web)의 보급 및 확산에는 HTML(Hyper Text Mark-up Language)이 중요한 역할을 수행해온 것이 사실이다. 그러나 HTML은 여러 가지 제약점을 갖고 있다. 즉 다양한 종류의 문서 논리 구조를 표현할 수 없으며, 또한 단순히 정보를 화면에 디스플레이하는 목적으로 만들어진 HTML은 컴퓨터간의 문서 정보를 상호 교환하는 전자 상거래나 어플리케이션 통합 등에 이용할 수가 없었다. 이것은 HTML의 DTD(Document Type Definition)가 하나로 고정되었기 때문이다.

이와 같이 다양한 전자 문서의 요구는 DTD를 정의할 수가 있는 메타 언어인 SGML(Standard Generalized Markup Language)로 해결할 수는 있으나, SGML 시스템 또는 어플리케이션의 복잡성 및 비경제성은 SGML의 확산에 장애가 되어 왔다.

이러한 HTML 및 SGML의 문제점을 해결하기 위한 배경으로 XML(eXtensible Markup Language)이 만들어 졌으며 97년에 W3C에 초안이 제안되어 현재 표준으로 공표되어 있다. XML은 메타 언어이면서 SGML을 인터넷에 적합하도록 단순화시킨 인터넷 버전이라 할 수가 있다[1-3].

일반적으로 XML은 단독으로 사용되기보다는 디스플레이의 제어와 스크립트의 기능을 갖고 있는 XSL (eXtensible Style Language), 관련 문서와 링크를 위한 XLL(XML Linking Language) 및 문서내의 엘리먼트 레벨의 처리를 가능케 하는 Xpointer(eXtended Pointer)등과 연계하여 이용되고 있다.

XML 문서가 SGML 문서와 다른 점은 DTD 없이도 문서의 제작과 배포가 가능하게 되었다는 것이다. DTD가 없는 XML 문서를 만들 때, W3C에서 제정한 최소한의 규약을 따라 작성된 XML문서를 Well-Formed XML 문서라고 한다[1, 3, 4].

현재 DTD를 포함하지 않는 Well-Formed XML 문서를 처리하는 도구 중 XLL 기능을 가지는 브라우저로는 Fujitsu의 HyBrick과 Link가 있으며 DTD 생성기는 SAXON의 DTD Generator와 Frontend가 있다. 그러나 이러한 프로그램들은 XLL기능을 가지는 브라우저와 DTD를 생성하는 기능을 독립적으로 가지고 있으며 각 기능들을 검증하는 단계에 있다[5].

본 연구에서는 클라이언트 상에서 Well-Formed XML 뷰어와 XLL 처리기, 자동 DTD 생성기를 모두 포함하는 Well-Formed XML처리 시스템을 구현하여 다음과 같은 문제점을 해결하기 위한 방안을 제안하고자 한다.

첫째, 기존 Well-Formed XML 브라우저에서 XLL의 확장 링크 기능 중 제한된 부분만을 제공하여 사용자들이 XLL의 확장된 기능을 효과적으로 사용하지 못하고 있으며

둘째, 기존 DTD 생성기에서는 하나의 Well-Formed XML 문서에 대해서만 DTD를 생성하여 등일 클래스의 유사한 논리구조를 가지는 많은 Well-Formed XML 문서들에서 추론된 보다 일반화된 DTD를 생성하지 못한다.

셋째, 웹 상에서 DTD를 포함하지 않는 Well-Formed 문서를 XLL 기능을 이용해 항해하면서 동시에 DTD를 생성하고자 할 경우, DTD를 발생시킬 문서를 별도로 저장한 후 DTD 생성기에서 DTD를 생성하는 비효율적인 구조이다.

이러한 문제점을 해결하기 위해 본 논문에서는 Well-Formed XML 뷰어와 XLL 처리기, 자동 DTD 생성기 Non-Validating 파서 등으로 구성된 시스템 환경을 제안하며, 본 시스템의 구성으로 DTD를 포함하지 않는 Well-Formed XML 문서에서 XLL의 확장된 링크 기능을 이용하여 편리하게 Well-Formed XML 문서를 항해할 수 있으며, 동시에 항해에 참여한 등일한 클래스의 Well-Formed XML 문서들의 논리구조에서 추론된 일반화된 DTD를 생성할 수 있다. 따라서 본 논문에서는 Xpointer 처리 및 확장링크 기능을 포함하는 XLL 처리기와 자동DTD 생성기의 설계 및 구현에 초점을 맞추었다

본 논문은 2장에서 관련 표준을, 3장에서는 시스템의 구성 및 구현, 4장에서는 실험 및 고찰, 5장에서는 결론 순으로 구성하였다

2. XML/XLL/XSL 등

2장에서는 XML, XLL 및 XSL의 일반적인 설명은 하지 않고 본 연구에 관련된 부분만을 간략하게 기술한다.

XML의 모든 문서는 XML 구문 규칙을 따르는 Well-Formed 문서이다. Well-Formed 문서 중에서

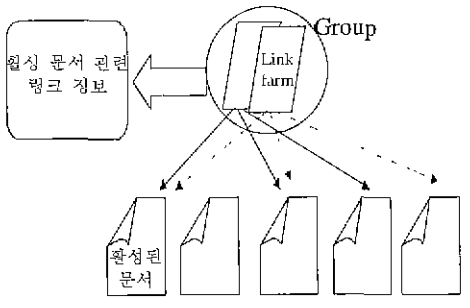
DTD를 포함하고 그 문서의 모든 엘리먼트가 해당 DTD 규칙을 따르는 문서를 유효한(Valid) 문서라고 한다[3, 4].

본 연구는 DTD를 갖지 않는 Well-Formed 문서를 대상으로 하였으며 이와 같은 XML문서의 조건을 나열하면 다음과 같다[1, 3] :

1) XML선언은 standalone 속성값이 yes이다
2) 공백 엘리먼트가 아닌 모든 엘리먼트는 시작 태그와 끝 태그를 생략할 수 없다.
3) 문서는 루트 엘리먼트로서 최상위 엘리먼트가 반드시 하나 존재하여야 한다
4) 엘리먼트는 올바르게 내포되어야만 한다.
5) DTD 없는 문서의 모든 엘리먼트는 속성을 가질 수 있다 그러나 속성을 지닐 수 있는 값은 CDATA로 제한된다
6) 데이터 내용 상에서는 '<'나 '&'와 같은 마크업 기호는 사용할 수 없다.

XLL은 HyTime(ISO 10744 · 1992)과 TEI(Text Encoding Initiative)의 연구 결과에 기반하여 구축되었으며 HTML의 앵커(Anchor)를 확장한 링크의 지정 및 의미 등을 규정하고 있다[6].

엘리먼트 중에서 링크 엘리먼트로 지정하기 위해서 예약된 속성 XML · LINK를 사용한다. 이 속성값으로는 SIMPLE, EXTENDED, LOCATOR, GROUP, DOCUMENT들이 있으며 이를 이용하여 확장 링크와 링크팜(LinkFarm)을 만들 수 있다 링크팜은 링크 정보들을 하나의 그룹으로 묶어서 한 장소에서 관리할 수 있는 기능을 제공한다. (그림 1)은 링크팜의 동작 개념을 나타내고 있다 링크의 수행을 제어하기 위하여 SHOW와 ACTUATE 같은 속성을 지정할 수가 있다.



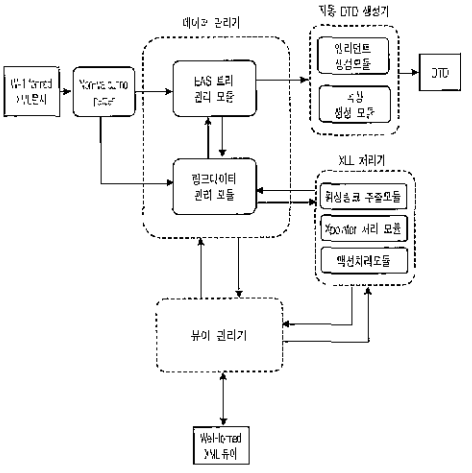
(그림 1) 링크팜의 동작

XPointer는 XML 문서의 일부분을 지정하기 위하여 URL의 일부로 사용하기 적합한 것이며, 목표로 하는 XML 문서 또는 문서의 일부를 지정하기 위하여 ROOT나 ID 등으로 구분할 수 있는 절대 위치를 시점으로 일련의 상대 위치 지정을 결합시켜 기술한다.

XSL은 XML 문서를 위한 스타일이나 스크립트를 기술하는 언어이다

3. 시스템의 구성 및 구현

본 연구의 전체 시스템 구성은 (그림 2)와 같다.



(그림 2) 시스템의 전체 구성도

웹에서 전달된 XML 문서는 우선 Non-Validating 파서에서 처리된다 Non-Validating 파서는 XML 문서가 DTD에 대해 유효한지 검사하지 않고 Well-Formed 문서 형식에 적합한지만을 검사한다. Non-Validating 파서는 일반적인 구조인 토큰생성기와 어휘분석기로 구성되며 XML 문서가 Well-Formed 문서인가를 검증한다 검증된 Well-Formed 문서는 구문 분석되고 논리 구조와 속성 및 스타일 정보를 포함하는 EAS (Element Attribute and Style) 트리를 EAS 트리 관리 모듈에서 생성하고 관리하며 링크 관련 정보는 링크데이터 관리 모듈에서 생성하고 관리한다.

뷰어 관리기는 EAS트리를 이용하여 Well-Formed XML 문서를 뷰어에 디스플레이한다. 본 시스템에서 사용되는 스타일은 CSS(Cascading Style Sheet)의 style 속성을 응용하였다. CSS의 style 속성의 값에는

font-size 등의 속성값들이 대응되나, 본 시스템에서는 31가지의 정형화된 스타일을 정의하여 스타일테이블에 저장하고 스타일테이블의 ID를 Style 속성값으로 사용하여 정의된 스타일을 지정한다. 즉, 본 스타일테이블의 ID를 style 속성의 속성값으로 사용하여 각 엘리먼트에 스타일을 부여한다.

XLL 처리기는 링크데이터 관리 모듈에서 링크정보를 제공 받아 확장 링크 및 링크팜을 처리하고 관리하며 Xpointer를 처리한다.

자동 DTD 생성기는 사용자가 DTD 생성을 요구한 경우, 본 시스템이 링크 항해를 수행하는 동안 동일 클래스의 각 문서에서 생성된 EAS 트리를 이용하여 DTD를 추론하고 이들로부터 일반화된 DTD를 생성하며, 생성된 결과를 파일로 저장한다.

3.1 데이터 관리기

데이터 관리기는 EAS트리 관리 모듈과 링크데이터 관리 모듈로 구성된다. EAS트리 관리 모듈은 EAS 트리 생성과 노드 삽입, 트리 순회 등을 관리한다. (그림 3)은 EAS 트리 노드의 실제 구조이다.

태 그 명
속 성 정 보
스타일 정보
링 크 정 보
실제 문서의 문자열
연결노드 정보

(그림 3) EAS 트리의 노드 구조

SEAS 트리 노드의 연결노드 정보는 부모노드, 자식노드, 이전 형제노드, 다음 형제노드의 4개의 노드로 구성된다. 연결노드 정보는 XLL 처리기에서 확장링크 및 링크팜에서 링크에 관련되는 문서의 논리구조 및 문자열을 탐색할 때, 문서 객체를 링크로 삽입시킬 때와 뷰어관리기가 문서를 디스플레이할 때 EAS 트리를 순회하게 된다 EAS트리 순회는 전위순회(Preorder Traversal) 방식이다.

스타일 정보에는 Well-formed XML 인스턴스에 포함된 태그의 style속성의 속성값이 대입된다. 스타일 정보는 스타일테이블의 ID로 사용된다. <표 1>은 스타일테이블에서 정의되는 스타일과 스타일값이다. Font와 Text의 스타일 종류에 대해 스타일 값을 조합하여 스타일을 정의하며 정의된 스타일은 스타일테이블의 ID로 지정할 수 있다.

〈표 1〉 스타일테이블에서 정의되는 스타일과 스타일값

스타일		스타일 값
ID		Integer
font	Family	Arial, TimeNewRoman, serif
	Style	Normal, italic
	Weight	Normal, bold
	Size	10, 12, 15, 18, 22, 25, 30
	Color	Black, green, blue, yellow, red
text	Indent	No, Yes
	Alignment	Left, middle, right
	Decoration	None, underline

링크데이터 관리 모듈은 확장링크 리스트들과 링크팜 트리로 구성된다.

(그림 4)는 확장링크 리스트의 노드 구조이며 (그림 5)는 링크팜 트리의 노드 구조이다.

엘리먼트 명	
속성 정보	
Xpointer 정보	
ID 정보	
Actuate 정보	
Show 정보	
Inline, Outofline 링크 정보	
연결노드 정보	이전노드
	다음노드

(그림 4) 확장링크 리스트의 노드 구조

엘리먼트 명
속성 정보
Xpointer 정보
ID 정보
Actuate 정보
Show 정보
연결노드 정보

(그림 5) 링크 팜 트리의 노드 구조

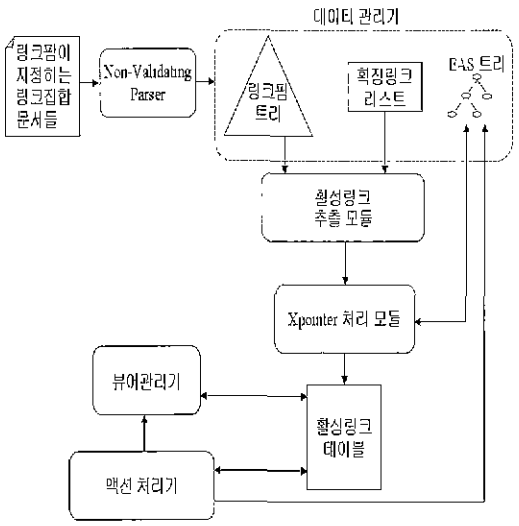
위 노드에서 속성정보에는 URL, Title, Role등 XLL에서 제정한 속성들이 포함된다. Actuate 정보는 목표 문서의 활성 시기를 결정하는 정보이며 Show 정보는 목표 문서의 활성 방식을 결정하는 정보이다 링크팜 트리의 노드는 EAS 트리의 연결노드 정보와 동일한 구조를 가지며 트리의 순회 방식은 전위순회이다.

EAS 트리에서 제공되는 링크 정보 중에서 문서 내의 확장링크 정보는 확장링크 노드를 생성하여 확장링

크 리스트에 저장되고, 링크팜 트리 노드는 링크팜이 지정하는 링크 집합 문서들로부터 링크 정보를 추출하여 생성되고 링크팜 트리에 저장된다.

3.2 XLL 처리기

XLL 처리기는 확장 링크 리스트와 링크팜 트리를 입력으로 받으며 활성링크 추출 모듈, Xpionter 처리 모듈, 액션 처리기로 구성된다. (그림 6)은 XLL 처리기의 구성이다.



(그림 6) XLL 처리기의 구조

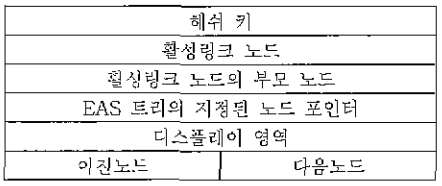
활성링크 추출 모듈은 활성된 문서와 관련된 확장 링크 및 링크팜의 링크 정보를 추출하는 모듈이다. 확장링크 리스트에서 Inline링크와 Outofline 링크 중 현재 활성된 문서와 관련된 모든 리스트의 노드를 추출하고, 또한 링크팜 트리에서도 활성된 문서와 관련된 링크팜 트리의 모든 노드를 추출한다. 이들 중 문서 내의 주소지정방식으로 Xpointer나 *ID*를 사용한 리스트는 Xpointer 처리 모듈에 입력된다.

Xpointer처리 모듈은 활성링크 추출 모듈에서 추출된 확장링크 리스트의 노드나 링크팜 트리의 노드에 포함된 Xpointer나 ID가 지정하는 문서 내의 지정된 위치를 EAS 트리를 순회하며 찾는다. Xpointer 의 처리는 각 Xpointer구문과 EAS 트리 관리기에서 제공하는 API 함수를 매핑하여 Xpointer가 지정하는 엘리먼트를 찾는다. <표 2>은 매핑 관계를 나타낸다.

(표 2) Xpointer 구문과 EAS트리 관리기의 API함수 매핑

Xpointer	API 함수
ID()	GetId()
ROOT()	GetRoot()
CHILD()	GetChild()
PSIBLING()	GetPSibling()
FSIBLING()	GetFSibling()
PRECEDING()	GetPreceding()
FOLLOWING	GetFollowing()

활성링크 테이블에는 Xpointer처리모듈 에서 추출된 활성된 문서의 링크팜이 지정하는 링크 목표들과 앵커들이 저장된다. 활성링크 테이블은 해쉬테이블을 이용한 리스트 구조이며 노드의 구조는 (그림 7)과 같다



(그림 7) 활성링크 테이블의 노드 구조

활성링크 테이블의 각 노드는 이전노드, 다음노드의 노드 연결정보에 의해 연결되며 이전노드와 다음노드에는 연결되는 노드의 해쉬키 값이 지장된다.

액션 처리기는 활성링크 테이블에 포함된 링크가 동작할 때 링크의 Show 속성과 Actuate 속성에 의해 동작하는 방식과 표현 방식을 관리한다. 본 시스템에서 <표 3>과 같은 방식으로 Show 속성과 Actuate 속성을 조합하여 링크의 동작 및 표현 방법을 제안한다

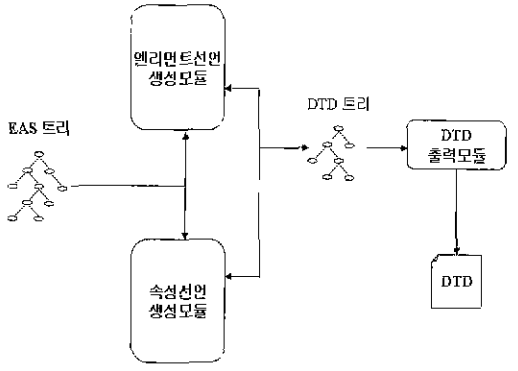
(표 3) Show 속성과 Actuate 속성의 조합

Show와 Acturate 조합	주석
Onload-embed	시스템이 목표 링크를 자동으로 삽입
OnRequest-replace	사용자가 앵커 클릭시 기존 XML 뷰어에 목표 리소스를 디스플레이
OnRequest-new	사용자가 앵커 클릭 시 새로운 XML 뷰어에 목표 리소스를 디스플레이

3.3 자동 DTD 생성기

자동 DTD 생성기는 본 시스템에 입력되는 Well-Formed XML 문서 중 같은 루트 엘리먼트를 가지는 동일 클래스의 각 문서들로부터 생성된 EAS 트리에서

Well-Formed XML문서의 각각의 엘리먼트 선언과 속성 선언을 생성하고, 생성된 선언들로부터 일반화된 DTD를 생성한다. (그림 8)은 자동 DTD 생성기의 구조이다.



(그림 8) 자동 DTD 생성기의 구조

자동 DTD 생성기의 내부 자료구조는 엘리먼트선언 노드로 연결된 DTD 트리와 DTD 트리의 각 엘리먼트 선언노드에서 지정하는 속성선언리스트로 구성되며 속성선언리스트는 속성선언노드로 연결되어 있다. (그림 9)은 엘리먼트선언노드의 구조이다.

엘리먼트 명
속성선언리스트의 포인트
엘리먼트 발생지시자
내용모델 발생지시자
연결노드 정보

(그림 9) 엘리먼트선언노드의 구조

엘리먼트선언노드에서 속성선언리스트의 포인트는 각 엘리먼트에 관련된 속성리스트를 지정하는 정보이며 엘리먼트 발생지시자는 각 엘리먼트선언노드가 엘리먼트선언의 내용 생성에 참여될 때 엘리먼트생성 모듈에서 생성된 발생지시자 정보가 저장된다. 내용모델 발생지시자는 엘리먼트 선언노드가 각 엘리먼트 선언의 내용모델 생성에 참여될 때 엘리먼트 생성모듈에서 생성된 내용모델의 발생지시자 정보를 저장한다. 연결노드정보는 DTD 트리의 부모노드, 자식노드, 이전노드, 다음노드 지정하는 정보로 구성된다.

(그림 10)은 속성선언리스트를 구성하는 속성선언노드의 구조이다.

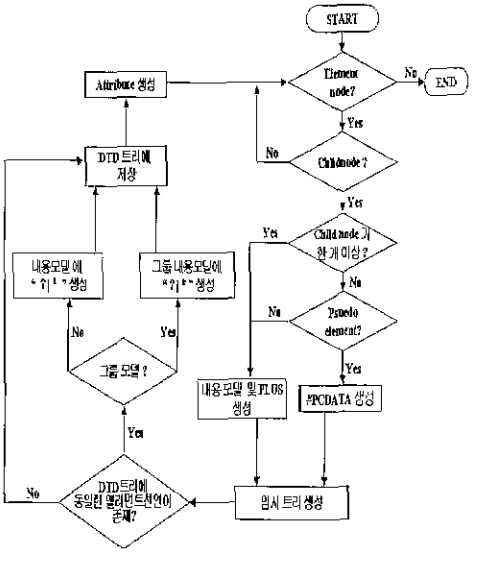
속성 명
선언된 값
발생횟수
디폴트값
FIXED값
연결노드 정보

(그림 10) 속성선언노드의 구조

속성선언노드에서 선언된 값은 CDATA나 ID값을 가지며 발생횟수는 동일한 엘리먼트 명에 포함된 속성의 디폴트값을 결정할 때 사용된다. 디폴트값에는 #FIXED, #IMPLIED, #REQUIRED 중에 하나가 저장되며 FIXED 값에는 디폴트값이 #FIXED인 경우 #FIXED 관련 정보가 저장된다. 연결노드 정보에는 이전노드와 다음노드로 구성되어 리스트에서 이전노드와 다음노드를 지정하는 정보가 저장된다

엘리먼트 생성모듈은 EAS트리의 노드를 입력으로 받아 엘리먼트 선언 부분을 생성하여 엘리먼트선언노드를 생성하고 엘리먼트선언노드를 DTD 트리에 연결한다. (그림 11)은 엘리먼트선언 생성모듈에서 제안한 엘리먼트선언 생성 알고리즘이다.

엘리먼트선언 생성 알고리즘에서는 EAS 트리에서 하나의 노드를 호출하고, 그 노드의 이름을 참조하여 엘리먼트형명을 생성한 후, 이 노드의 자식 노드들을 찾는다.



(그림 11) 엘리먼트선언 생성 알고리즘

자식 노드가 발견되는 경우, 각 노드들을 내용 모델로 만들기 위해 엘리먼트선언노드를 생성하고 그 내용의 발생 순서로 엘리먼트선언노드를 연결하여 임시 트리를 구성한다. 또한 연결자와 발생지시자를 각 엘리먼트선언노드에 대하여 생성시킨다.

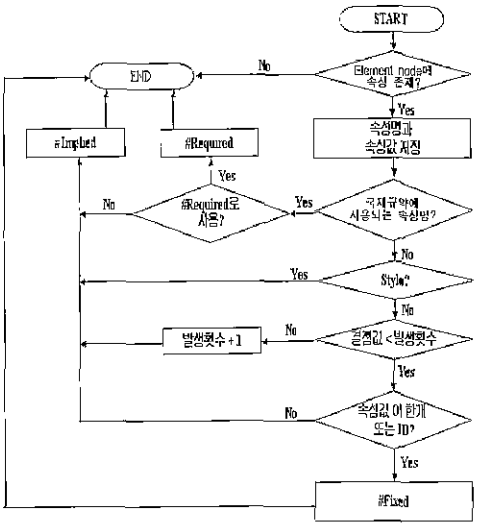
자식 노드가 발견되지 않았을 경우는 그 엘리먼트가 공백 엘리먼트인지 또는 구문 분석된 데이터(#PCDATA)인지를 판별하여 엘리먼트선언노드를 생성한다. 이러한 과정을 통해 부모노드와 자식노드만으로 구성된 임시 엘리먼트선언 트리를 생성한다.

하나의 임시 엘리먼트선언 트리 생성 후, 기존에 만들어진 DTD 트리에서 동일한 이름의 엘리먼트 선언이 있는지 검색하고, 존재하지 않을 경우 임시 엘리먼트선언 트리를 DTD 트리에 연결한다. 그러나 동일한 이름의 엘리먼트 선언이 존재하는 경우는 DTD 트리의 엘리먼트 선언 내용 모델과 비교하여 내용 모델에 연결자 OR와 발생지시자 REP, OPT, PLUS를 생성하여 현재까지 비교된 내용모델을 포함하는 일반화된 내용모델을 생성한다. <표 4>은 본 시스템의 연결자와 발생지시자 생성 조건이다.

〈표 4〉 연결자와 발생지시자 생성 조건

OR (|) 연결자 생성 조건	EAS 트리 상에서 1개 이상의 동일한 이름의 부모노드에 포함된 서로 다른 엘리먼트 명을 가지는 자식노드가 단일하게 존재할 경우
OPT (?) 발생지시자 생성 조건	EAS 트리 상에서 2개 이상의 동일한 이름의 부모노드에 포함된 동일한 엘리먼트 명을 가지는 자식노드가 존재하면서 동일한 이름의 다른 부모노드에서는 존재하지 않을 경우
PLUS (+) 발생지시자 생성 조건	EAS 트리 상에서 2개 이상의 동일한 이름의 부모노드에 포함된 동일한 엘리먼트 명을 가지는 자식노드가 한 번 이상 존재할 경우
REP (*) 발생지시자 생성 조건	EAS 트리 상에서 2개 이상의 동일한 이름의 부모노드에 포함된 동일한 엘리먼트 명을 가지는 자식노드가 한 번 이상 존재하면서 동일한 이름의 다른 부모노드에서는 존재하지 않을 경우

속성선언 생성모듈은 각각의 엘리먼트 선언이 생성된 후 EAS 트리의 노드에서 지정하는 속성리스트를 입력으로 받아 속성선언리스트의 속성선언노드를 생성한다. (그림 12)은 속성선언 생성모듈에서 제안한 속성선언생성 알고리즘이다.



(그림 12) 속성선언생성 알고리즘

엘리먼트선언 생성이 종료되면 DTD 트리의 엘리먼트 노드가 속성을 가지고 있는 경우에만 속성 선언 알고리즘을 실행한다. 먼저 속성 이름을 생성하고, 속성 이름에 대입되었던 속성값을 이용하여 디폴트값을 생성한다. 디폴트 값을 결정하는 경계값은 사용자가 설정한다. 속성타입은 Well-Formed XML 문서에서는 CDATA만 사용되므로 속성타입은 CDATA 고정시켰다.

최초로 속성 선언이 발생될 때는 디폴트값은 #IMPLIED를 기본값으로 한다. 기존에 속성 선언이 만들어져 있는 경우는 그 속성이 몇 번 비교되었는지를 알려주는 속성선언노드의 발생횟수값과 사용자가 설정한 경계값을 비교하여 발생횟수값이 큰 경우에는 <표 5>에 따라 디폴트 값이 결정된다.

〈표 5〉 속성선언 생성모듈의 디폴트값 생성

디폴트값	조 건 식
#FIXED	한 엘리먼트에 대해서 하나의 속성과 하나의 속성값만을 가지는 경우
#IMPLED	한 엘리먼트에 한 개 이상의 속성이 존재하고 각각의 속성에 대해서 두 개 이상의 속성값이 존재할 경우
#REQUIRED	한 엘리먼트에 한 개 이상의 속성이 존재하고 각각의 속성에 대해서 하나의 속성값만이 존재할 경우

인스턴스에 포함된 CSS 형식을 응용한 style 속성은 Well-Formed 문서의 논리구조의 속성에 포함되기보다는 부가적인 정보로 처리한다. 본 시스템에서는

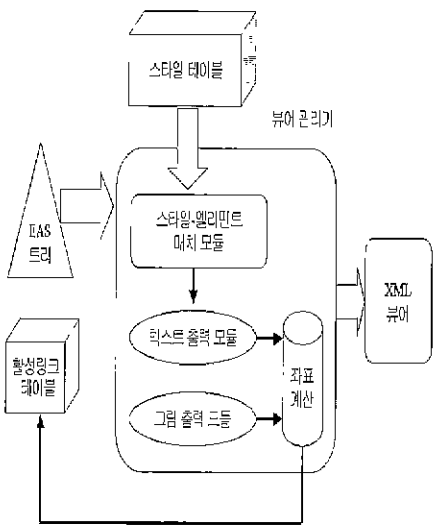
각 엘리먼트에 style 속성을 사용하여 시스템에서 정의된 스타일을 지정하는 방식을 취하여 다른 Well-Formed 문서에서의 스타일 정보와 상이하다. 따라서 속성선언 생성 알고리즘에서는 본 시스템의style속성의 디폴트값을 #IMPLIED로 적용되도록 가정하였다

DTD 출력모듈은 엘리먼트선언 생성모듈과 속성선언 생성 모듈에 의해 생성된 DTD 트리를 순회하며 사용자의 요청에 따라 DTD 문서를 생성하거나 뷰어에 디스플레이할 수 있으며, 사용자가 DTD를 추가/수정할 필요가 있을 경우, 저장된 DTD를 변경할 수 있다.

3.4 뷰어 관리기

뷰어 관리기는 EAS 트리의 각 노드의 엘리먼트 정보와 스타일 정보를 이용하여 XML 뷰어에 문자열과 그림을 출력한다. (그림 13)은 뷰어관리기의 구조이다.

본 시스템에서 정의한 스타일테이블에는 정형화된 스타일들을 포함한다. <표 1>에 있는 스타일과 각각의 스타일값을 조합하여 스타일테이블의 ID가 지정하는 각각의 스타일을 정의한다 <표 6>는 스타일 테이블에 정의된 스타일이다



(그림 13) 뷰어 관리기의 구조

뷰어관리기는 스타일-엘리먼트 매치 모듈, 텍스트 출력 모듈과 그림 출력 모듈로 구성된다. 스타일-엘리먼트 매치 모듈은 EAS트리에 있는 각 노드의 스타일 정보(style의 속성값)가 존재하는 경우, 스타일 정보를 스타일 테이블의 ID로 이용하여 스타일테이블의 ID가 지정하는 스타일 정보를 읽어 텍스트출력모듈이나 그림출력모듈에 전달하며 엘리먼트 속성 중 style이 존재하지 않는 경우는 스타일 테이블의 1번 ID에 대응되는 스타일 정보를 텍스트출력모듈에 전달한다.

텍스트출력모듈과 그림출력모듈은 스타일-엘리먼트 매치모듈에서 입력된 스타일 정보를 이용하여 문자열 및 그림의 스타일을 결정하고 또한 문자열과 그림이 출력될 좌표를 계산하여 XML 뷰어에 문자열과 그림을 출력하고 활성링크 테이블의 디스플레이영역에 출력 좌표값을 저장한다

〈표 6〉 스타일테이블에 정의된 스타일

스타일		스타일 값				
ID		1	2	·····	30	31
font	Family	Arial	Arial	·····	Serif	Serif
	Style	Normal	Normal	·····	Normal	Italic
	Weight	Normal	Normal	·····	Bold	Normal
	Size	10	12	·····	18	15
	Color	Black	Black		Green	Black
text	Indent	No	Yes	·····	No	No
	Ahgn-Ment	Left	Left	·····	Middle	Middle
	Deco-+Ration	None	None	·····	None	Under-line

4. 실험 및 고찰

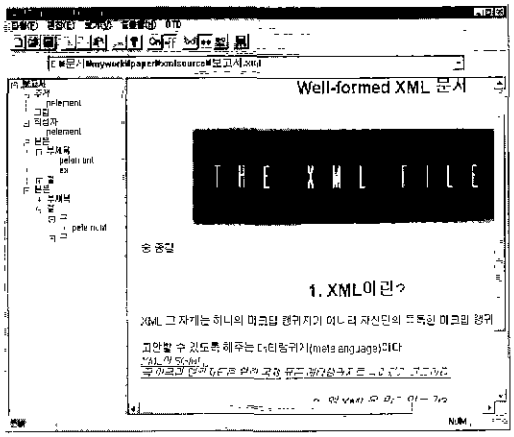
4.1 실 험

인터넷 환경에서 본 시스템을 사용하여 Well-Formed XML 문서를 XML 뷰어에 디스플레이함과 동시에 XLL링크의 등작 및 자동 DTD 생성기를 실험하였다.

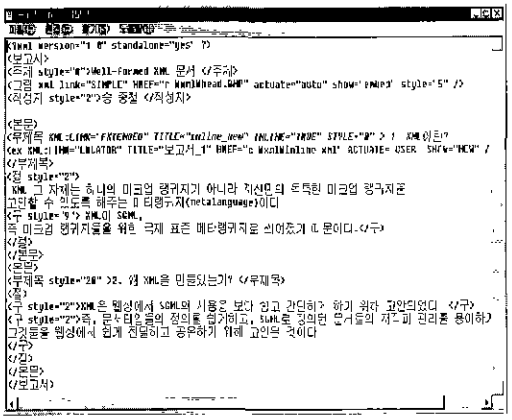
펜티움 133Mhz 32Mbyte 주메모리 시스템에서 운영체제는 Win98로 하였으며 Visual C++ 5.0을 이용하여 데이터퀴리기, XLL처리기, 자동 DTD 생성기, 뷰어 관리기들을 클래스로 구현하였다.

(그림 14)은 본 시스템에서 DTD를 포함하지 않는 Well-Formed XML문서를 처리하여 출력하는 XML 뷰어를 보이고 있다 왼쪽 페인(Pane)은 사용자들이 문서의 논리 구조에 좀 더 친숙하도록 활성된 문서의 논리 구조를 트리 형태로 보여주고 있으며, 오른쪽 페인은 XML문서를 브라우징한 결과이다.

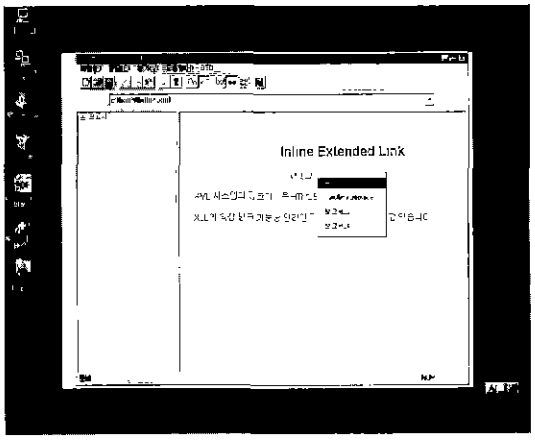
(그림 15)은 (그림 14)에서 브라우징된 Well-Formed XML문서의 인스턴스이다. 본 인스턴스에서 style 속성은 시스템에 정의된 스타일테이블의 ID를 속성값으로 가진다.



(그림 14) Well-Formed 문서를 출력한 결과



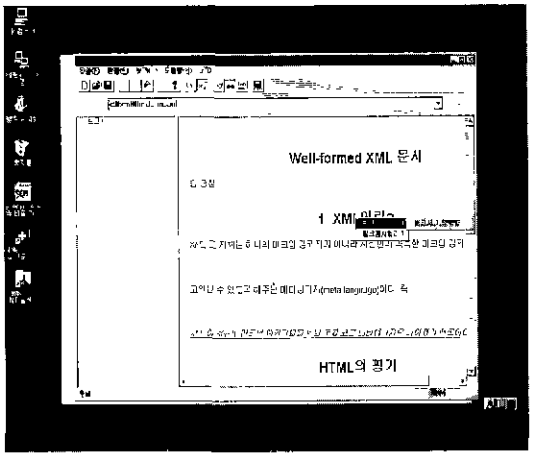
(그림 15) 브라우즈된 Well-Formed XML문서의 인스턴스



(그림 16) 확장 링크 중 아웃어브라인 링크의 동작

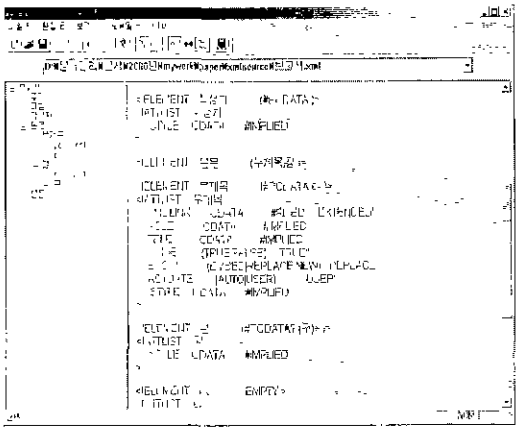
(그림 16)는 확장 링크 중 다방향 및 양방향 링크가 가능한 아웃어브라인 링크의 동작 화면이다.

(그림 17)은 XLL의 확장된 링크 기능 중 링크들을 그룹으로 묶어서 허브(hub) 형식으로 관리 할 수 있는 링크 팜의 동작 화면이다.



(그림 17) 링크 팜의 동작

(그림 18) 는 Well-Formed 문서들로부터 일반화된 DTD를 자동 생성한 결과이다.



(그림 18) 자동 생성된 DTD

4.2 고 찰

본 논문에서 기술한 연구는 클라이언트 상의 Well-Formed XML 문서 처리 시스템은 XLL의 확장 링크, 링크팜의 구현과 DTD의 자동 생성에 중점을 두었다.

〈표 7〉 본 시스템과 기존의 Well-Formed XML 문서 처리 시스템 기능 비교

기능 구분		본 논문의 시스템	Well-Formed XML 관련 시스템	
			Fujitsu의HyBrick	Frontend
시스템의XLL기능과 DTD생성 기능 포함		통합된 클래스로 제공	XLL만 처리	DTD 생성기능만제공
XLL 기능		확장링크 및 링크팜 기능 제공	Onload-embed 기능 제공 안함	기능 없음
DTD 생성 기능	DTD 생성시 참조되는 문서 수	1개 이상의 문서에서 DTD를 추출, 생성	기능 없음	단일문서에서 DTD 생성
	내용모델의 발생지시자 생성 기능	?, *, + 기능	기능 없음	기능 없음
	내용모델 처리 수준	복잡한 내용모델 처리 기능	기능 없음	단순한 내용모델 처리

따라서 본 연구의 파서도 연구 목적에 적합하도록 Non-Validating 파서로 구현하여 내장 엔티티는 처리할 수 있으나 일반 엔티티는 처리할 수 없도록 단순화시켰다. 또한 문서의 스타일도 CSS의 style 속성을 응용하여 단순하시켰다.

본 시스템에서 링크의 표시는 마우스 모양의 변화로 사용자가 앵커를 인식할 수 있게 하였으며 주소지정방식은 Xpointer의 기능 중 ID, ROOT, CHILD, PSIBLING, FSIBLING, PRECEDING, FOLLOWING을 이용하여 문서의 논리구조 및 문자열을 지정할 수 있도록 하였으며, 다른 Xpointer 기능들도 확장할 예정이다. 링크의 Show와 Actuate 기능들은 모두 구현되었으며, 문서 객체를 자동 삽입을 위하여 Onload-embed 기능을 구현하였다.

본 연구에서 제시한 Well-Formed 문서 처리 시스템과 기존 Well-Formed 문서를 처리 시스템과 기능을 비교하면 <표 7>와 같다.

5. 결 론

본 연구에서는 클라이언트 상에서 DTD를 포함하지 않는 Well-Formed XML 문서를 처리하기 위한 시스템을 설계 및 구현하였다 XML 뷰어는 스타일 처리 시 제한된 기능으로 구현하였고 연구의 초점을 일반화된 DTD의 자동 생성, XLL의 확장 링크 및 링크팜의 구현에 두었다.

본 시스템의 구현으로 XLL 기능의 유효성을 확인하였고 XLL의 하이퍼링크 기능을 이용하여 Well-Formed XML 문서를 순회하는 동안, 동일한 클래스의 Well-Formed XML 문서들로부터 DTD를 추출하고 이들로부터 일반화된 DTD를 생성할 수 있었다. 구현 환경은 Win98 하에서 Visual C++ 5.0을 이용하였다.

XLL의 확장 링크 기능을 구현하기 위하여 링크 정보를 중앙에서 관리하기 위한 링크팜을 구축하였으며, 링크팜에는 다방향 링크, 양방향 링크, 리소스와 분리된 링크 등을 처리하기위한 정보에 집중시켰으며, 또한 링크 종단이 포함된 문서를 별도로 관리함으로써 링크 정보의 관리를 용이하게 할 수 있었다. 특히, 링크를 운행할 때 링크 팜으로 부터 활성화된 XML 문서에 해당하는 링크 정보를 동적으로 관리함으로써 필요한 정보를 보다 용이하게 찾을 수가 있었다.

동일한 클래스에 속하는 문서들로부터 DTD를 자동 생성하는 방식을 연구하였으나 모든 클래스를 대상으로 하는 자동 생성 알고리즘까지는 구현하지 못하였으며 자동 생성된 DTD를 사용자가 직접 개입하여 검증 또는 변경할 수 있도록 고려하였다.

본 논문에서 제시한 시스템의 구현에는 다음과 같은 제한 사항을 두었다.

- 스타일 정보는 CSS의 style 속성 형태를 취하였으나 정형화된 스타일을 스타일테이블에 미리 정의하여 사용하였다. 따라서 XSL을 이용하는 보편적인 방법으로 스타일을 제어할 수 있는 연구가 계속되어야 한다.
- 자동 DTD 생성기에서 DTD를 발생시킬 때 내용모델 그룹이 셋 이상 중첩될 경우는 처리할 수 없다. 또한 비교적 단순한 OR 구조를 처리 대상으로 하였다.
- 일반 엔티티의 처리를 포함하지 않는다

앞으로의 연구 및 개선 방향은 좀 더 복잡한 DTD를 생성하기 위해 자동 DTD 생성기를 OR 구조와 둘 이상의 내용 모델 그룹의 중첩, Namespace를 처리할

수 있도록 개선하여야 한다. 또한 일반 엔티티의 효과적인 처리를 위하여 Repository에 관한 연구와 XSL및 XSLT에 관한 연구도 계속하여야 할 것이다.

참 고 문 헌

[1] http : //www.sil.org/sgml/xml.html

[2] ISO 8879 : 1986 Information Processing-Text and office systems-Standard Generalized Markup Language (SGML), Geneva, 15 October 1986.

[3] Dan Connolly and Jon Bosak, Extension Markup Language(XML), 1997, http : //www.w3c.org/XML

[4] 정회경, "XML 가이드", 그린 1998, pp.24-30.

[5] http : //www.oasis-open.org/cover/xml.html

[6] Tim Bray and C.M.Sperberg-McQueen, "Extensible Markup Language(XML) : Part I, Syntax, W3C Working Draft," 1997, 6, http : //www.w3.org/TR/WD-xml-lang.html.

[7] Tim Bray and Steve DeRose, "Extensible Markup Language(XML) : Part II, Syntax, W3C Working Draft," 1997, 6, http : //www.w3.org/TR/WD-xml-link.html

[8] Natanya Pitts-Moultis, Cheryl Kirk, "Black Book," CORIOLIS, 1999

송 종 철

e-mail : jcsong@etri.re.kr

1999년 광운대학교 대학원 컴퓨터공학과(공학석사)

1999년~현재 한국전자통신연구원 연구원으로 재직중

관심분야 : XML, SGML, EDI, 지식관리시스템 등

문 병 주

e-mail : bjmoon@info1.etri.re.kr

1990년 부산대학교 전자공학과 졸업(학사)

2000년 충북대학교 대학원 전산학과 졸업(이학석사)

1991년~현재 한국전자통신연구원 정보유통연구팀 재직중

관심분야 : 정보검색, 지식관리시스템 등

홍 기 채

e-mail : gchong@etri.re.kr

1993년 충남대학교 대학원 전산학과 졸업(이학석사)

2000년 충남대학교 대학원 컴퓨터과학과 재학(이학박사 과정)

1984년~현재 한국전자통신연구원 정보유통연구팀 재직중

관심분야 : 데이타베이스시스템, 정보검색, 지식관리시스템 등

정 현 수

e-mail : hsjung@info1.etri.re.kr

1990년 숭실대학교 대학원 전산학과 졸업(공학석사)

1995년 숭실대학교 대학원 전산학과 졸업 (공학박사)

1984년~현재 한국전자통신연구원 정보유통연구팀 팀장 재직중

관심분야 : 정보검색,지식관리시스템,XML 등

김 규 태

e-mail : ivorykim@explore.kwangwoon.ac.kr

1996년 광운대학교 대학원 컴퓨터공학과(공학석사)

1996년~현재 광운대학교 대학원 컴퓨터공학과(공학박사과정)

관심분야 : SGML/XML,데이터베이스 등

이 수 연

email : leesy@daisy.kwangwoon.ac.kr

1977년 연세대학교 전자공학과 (공학석사)

1983년 일본 교토대학교 정보공학과 (공학박사)

1973년~현재 광운대학교 컴퓨터공학과 정교수

관심분야 : SGML/XML, IETM, 전자상거래 등