XSL-FO 문서를 PostScript Format으로 변환하기 위한 PostScript-Converter에 관한 연구

유동석* · 김차종**

A Study on PostScript-Converter for conversion XSL-FO into PostScript Format

Dong-Sok Luy* · Cha-Jong Kim**

요 약

현재의 전자문서처리 환경은 WYSIWYG 방식이다. 이를 위해 문서를 논리적인 구조와 물리적인 구조로 구조화하였고 이러한 구조를 마크업언어로 표현하고 있다. 특히 인터넷상의 전자문서 작성 및 교환을 위한 마크업언어로 XML이 발표되어 전자문서의 표현과 같은 전통적인 사용에서부터 검색을 위한 데이터베이스화에 이르기까지 전자문서의 활용 영역이 다양해지고 있다. 그러나 출력 품질 면에서 워드프로세서나 전자출판에 의한 전자문서와 XML 문서의 출력 품질은 매우 큰 차이가 있다. 이는 비록 XML 문서가 스타일 정보를 포함하고 있긴 하지만 화면 출력과 인쇄 매체로의 출력 모두 고품질의 출력을 위한 적용이 부족했기 때문이다. 이러한 문제 해결을 위해 W3C에서는 고품질의 XML 출력 문서를 얻을 수 있도록 XSL-FO(XSL-Formatting Object)를 개발 하였다. 한편 고품질의 전자출판물을 얻기 위해 페이지 기술 언어(PDL)가 필요하고, 이의 업계표준인 PostScript가 이미 널리 사용되고 있다. 따라서 본 논문에서는 XML-FO를 PostScript에 적용함으로써 고품질의 XML 출력문서를 얻기 위한 변환기를 설계하였다.

ABSTRACT

At present, the electronic document is being processed in WYSWYG mode. For this, a document is structured by the logical structure and the physical structure, and is presented by the markup language. After XML is announced, an application scope of the electronic document is extended from interchanging to searching. However, in point of output quality, a XML document image on a browser has lower quality than a general document image on desktop publishing. The reason is which output function of a browser has not capability for high quality printing. The W3C developed XSL-FO(XSL-formatting Object) for style sheet formatting and PDL(Page Description Language) as like PostScript is already developed and used widely. In this paper, we designed the PostScript-Converter to get a high quality document image by converting XSL-FO into PostScript format.

키워드

XSL-FO, PostScript, convert, document, page layout

*국립한밭대학교 정보통신전문대학원 컴퓨터공학과　　**국립한밭대학교 정보통신·컴퓨터공학부 교수

접수일자 : 2004. 1. 17

Ⅰ. 서 론

1.1 본 연구의 배경 및 목적

컴퓨터의 기술이 발달됨에 따라 컴퓨터를 이용한 전자문서의 처리 환경이 용이하게 되었고, 이를 위해 간단한 워드프로세서(word processor)에서부터 전문지식 시스템에 이르기까지 많은 문서작성 시스템이 사용되고 있다. 특히 인터넷상의 전자문서 작성 및 교환을 위한 마크업언어로 XML이 발표되어 전자문서의 활용 영역이 다양해지고 있다. 그러나 출력 품질 면에서 워드프로세서나 전자출판 제작 시스템에 의한 전자문서와 XML 문서의 출력 품질은 매우 큰 차이가 있기 때문에 이를 해결하기 위해서 보다 효과적인 XML문서 출력 시스템이 요구되고 있다.

이런 문제점들은 XML문서 출력의 표준화를 가속시켰고, 새로운 표준인 XSL-FO1.0이 W3C에 의해 제정되었다. XSL-FO는 XML문서의 내용을 종이와 같은 매체에 효과적으로 인쇄하기 위해 사용되는 것으로 문서 스타일의 포맷팅 정보를 이용하여 다양한 출력 환경을 구축할 수 있다.

한편 출력 문서의 품질은 어떠한 프린터를 사용하는지 그리고 얼마나 다양한 폰트를 지원하느냐에 따라 다르게 나타난다[1]. 프린터에는 크게 충격형 프린터와 비충격형 프린터로 나눌 수 있는데 충격형 프린터(Impact printer)를 사용하지 않고 레이저 빔 프린터와 같이 프린팅 디바이스에 프로세서를 탑재한 프린팅 디바이스가 개발되면서 페이지를 기술하는 형식으로 인쇄가 이루어진다. 레이저 빔 프린터는 서로 다른 활자와 서체를 조작할 수 있고 그래픽스와 텍스트를 혼용할 수 있는 출력장치이다. 이것은 내부에 RIP(Raster Image Processor)라는 프로세서가 있어 페이지 기술에 관한 정보를 번역하여 비트맵을 발생 시키고 있다[2]. 따라서 에디터 또는 포맷터 등과 같은 텍스트 지향적인 시스템으로부터 만들어진 페이지 또는 문서를 레이저 빔 프린터와 같은 출력장치로 출력하기 위해서는 페이지의 기술을 프린터에게 알려 주어야 한다[3]. 예를 들면 행 간격, 폰트 종류, 폰트 크기 등의 지정을 프린터에 알려 주어야 하고, 이를 위한 언어가 페이지 기술언어(Page Description Language : PDL)이다. 바꿔 말하면 출력장치와 텍스트 지향 시스템 간의 기술 용어(description terminology)를 표준화 시키는 역할을 하는 언어이다. 페이지 기술언어에는 PostScript(Adobe Systems), Interpress (Xerox), Document Description Language(Imagen)가 있는데 대부분 포스트스크립트를 채택하고 있다[4],[5]. 이유는 포스트스크립트의 사양이 공개되어 있고 Apple사의 LaserWriter와 같이 저 가격의 PC용 레이저 프린터에 채용 가능하며, 스택지향 언어로서 저장 공간을 관리할 수 있고, 프로그래밍적 능력 또한 충분하기 때문이다.[1]

PostScript언어로 구성된 파일은 프린터 장치(output device)에 상관없이, 컴퓨터의 플랫폼에 관계없이 문서를 프린트 할 수가 있다. 또한 이 파일은 Vector 그래픽 형식으로 확대나 축소 시에도 그림의 질이 일정하다는 장점을 가지고 있어서 PostScript는 문서가 정확하게 프린팅 되게 하고, 보다 완성도 높은 성능을 가질 수 있게 한다.[6]

따라서 본 논문에서는 인터넷상에서 사용되고 있는 XML문서를 고품질로 인쇄하기 위해 XSL-FO 즉, 문서 스타일의 포맷팅 정보를 PostScript에 적용하기 위한 변환기를 설계하였다.

1.2 기존 시스템의 연구

XSL-FO를 기반으로 하는 대표적인 문서 변환 시스템은 FOP (Formatting Objects Processor)이다. FOP는 XSL-FO에 의해 유도되는 세계 최초의 프린트 포맷터이고, 출력장치에 무관한 포맷터이다. 현재 지원되는 포맷으로는 PDF, PCL, SVG, XML, Print, AWT, MIF, TXT등이 있다. FOP의 가장 마지막 버전은 FOP[0.20.5]이고, 이것은 XSL-FO Version 1.0 W3C Recommendation의 일부분을 구현 하였다[7].

Antenna House XSL-FO V2는 XSL-FO Version 1.0 W3C Recommendation을 따른다. GUI 환경에서 XML을 브라우징하고, XML을 인쇄하며, XML 문서를 PDF로 변환시킨다. V2.5는 확장된 프로퍼티들을 제공하고, 전 세계 주요 50개국의 언어로 포맷팅이 가능하다. 포맷팅 엔진 V3는 Scratch로부터 개발되었고, 긴 문서의 포맷팅을 할 수 있다[8].

Xinc는 속도와 메모리를 효율적으로 사용하도록 디자인된 XSL-FO프로세서이다. Xinc는 Java API를 통한 서버 컴포넌트에서 사용되고 있고, COM 인터페이스를 사용하는 마이크로소프트 서버 환경에서도 사용할 수 있다[9].

이들 프로세서들은 온라인에서 기능을 제공하기도 하고, 각각의 엔진이 탑재된 서버 환경에서 사용되기도 하고, 또한 높은 품질을 제공한다는 장점을 가지고 있는 반면 이들을 사용하기 위해서는 라이센스가 필요하다는 단점이 있다.

1.3 본 논문의 내용 및 구성

본 연구는 XSL-FO로 기술된 문서를 입력 받아서 PostScript 언어로 변환 시키는 것을 목적으로 한다. 입력받은 문서를 DOM파서(parser)로 파싱

(parsing)을 한다. DOM을 사용한 이유는 다음 3가지이다. 첫째는 애플리케이션을 개발할 때 XML의 적용이 요구되는 시점은 대량의 데이터가 아닌 특정 부분에서 의미 있는 데이터를 접근하기 위한 것이기 때문에 시스템의 용량을 초과하거나 DOM구성으로 인한 시간의 지체는 많지 않다. 둘째로 한 번 DOM 트리가 구성되면 메모리 영역 안에 XML 데이터 구조가 남아있기 때문에 언제라도 XML 데이터 중 원하는 부분에 접근할 수 있다. 셋째는 DOM을 이용하면 손쉽게 XML 데이터를 검색하고 필요한 경우 수정, 추가, 삭제할 수 있기 때문이다[10].

XSL-FO의 문서 내의 오브젝트와 프로퍼티들을 PostScript의 명령어로 변환된 파일을 텍스트 창에 출력하고, PostScript 인터프린터가 내장된 프린터를 불러와서 변환된 문서를 인쇄 하고, 파일로 저장하도록 하였다.

II. 관련기술

2.1.XSL

XSL(extensible Stylesheet Language)은 'XML의 내용과 표현의 분리'라는 중요한 의미를 부여해 주는 역할을 담당하고 있는 XML 기술 중의 하나이다. XSL은 1997년 개발되기 시작했으며 SGML의 DSSSL과 HTML의 CSS에 기반하고 있다. 이 기술은 XML스펙을 따르고 있으므로 XML과 같은 장점을 가진다. XML문서의 구조를 재구성하거나, 데이터를 추가하고, 프레젠테이션용으로 변환시키기 위해 사용한다. XSL은 크게 두 가지 기술로 나누어지는데 하나는 XSL-FO이며, 나머지는 XSLT이다. XSL-FO는 XML문서의 시각적 표현을 위해 사용되는 포맷팅 언어이고, XSLT는 XML문서를 다른 형태의 문서로 변환하기 위해 사용되는 변환 언어이다[11].

2.2 DOM & SAX

DOM은 HTML이나 XML과 같은 문서를 관리하기 위한 프로그램 API로서, 파서에 의해 파싱된 문서를 메모리 내에서 트리 형태의 계층 구조로 저장하여 문서 내의 어떤 부분도 프로그래머가 자유자재로 검색할 수 있게 해 주고, 특정 요소를 문서의 어느 곳에든 삽입, 수정, 삭제할 수 있게 해 준다. XML문서를 처리하는 데에는 DOM과 SAX 두 가지 방법이 있는데, 그중 DOM은 W3C가 공식적으로 인정하는 표준으로 XML문서 처리를 위해 다양하게 사용된다[12].

문서 트리에서 각 노드들은 문서의 논리적인 구조를 의미하며, 트리의 모든 노드들은 Node라는 클래스로부터 상속받는다. 즉 문서의 논리적 구조는 Node클래스로부터 상속받은 다양한 타입의 클래스들로 구성된다. 예를 들어, 문서의 원소는 Node 클래스로부터 상속받은 Element 클래스로 표현되고, 텍스트는 Node의 자손인 Text클래스로 표현된다.

J2SDK에는 DOM을 지원하기 위해 javax.xml.parser패키지와 org.w3c.dom패키지를 포함하고 있다. javax.xml.parser패키지는 XML문서를 파싱하고 Document객체에 저장시켜 주기 위한 DOM파서가 내장되어 있고, org.w3c.dom 패키지에는 DOM Level2 Core에 근간하여 XML문서의 각 부분에 해당하는 인터페이스가 포함되어 있다.

DOM 트리에서 어떤 노드 타입은 다른 노드 타입들을 자식으로 가질 수 있고, 다른 어떤 노드 타입은 자식 노드를 갖지 못할 수 있다[6].

2.3 PostScript 언어

PostScript란 'Adobe Systems Ltd.'에서 개발한 인쇄용 언어로, PostScript 파일은 Vector 그래픽 형식으로 되어 있어 확대나 축소할 때 그림의 질이 일정하다는 장점이 있다. PostScript 파일은 PostScript 폰트가 내장된 프린터에서는 파일은 lpr이나 copy 명령어를 이용하여(예 : copy test.ps lpt1) 출력이 가능하나, PostScript 폰트가 내장되지 않은 대부분의 프린터에서는 GhostScript 등 인터프린터를 이용하여 Image로 변환 과정을 거친 뒤 , 인쇄가 가능하다. 현재로서는 GMT에서의 한글 구현은 지원되지 않기 때문에 생성된 PostScript 파일을 Photoshop, Gorel Draw 및 Illustrator 등의 Graphic Software를 이용하여 편집하여야 한다. 여기서 PostScript언어 전체를 기술하기에는 너무나 양이 방대하기 때문에 본 논문과 관련된 언어 특징과 사양 그리고 사전에 관해서만 설명하기로 한다[13].

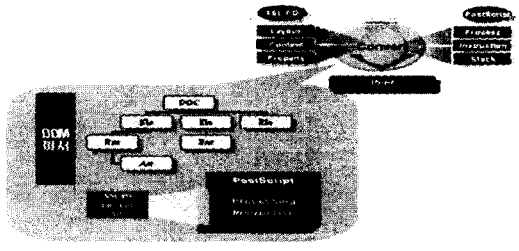
III. 시스템 개요 및 구성

3.1 시스템 개요

본 시스템은 다음 그림 1.과 같이 전자문서처리 환경에서 현재 XML 문서가 가지고 있는 스타일 정보를 이용하여 고품질의 XML 출력문서를 얻을 수 있는 시스템이다. 이 시스템은 문서를 파싱하기 위해 DOM 파서인 xerces-1.4.1, 구조화 시키고 변

환하기 위해 DOM API를, 변환을 위해 J2SDK1.4를 사용하였다.

그림 .1 시스템 개요
Fig. 1 Abstract of system



3.2 시스템 구성

그림 2.는 시스템의 전체 구성을 나타낸 것으로 본 시스템은 파일관리 모듈, 트리 View 모듈, Editing 모듈, Converter, Printing 모듈 등 5개 모듈로 구성된다.

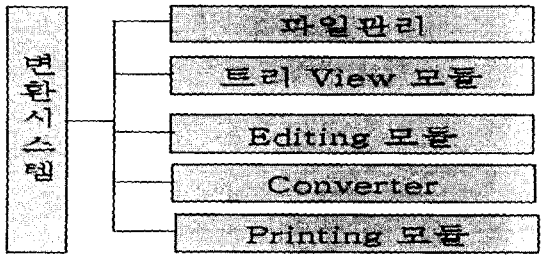


그림 2. 시스템 구성
Fig.2 Organization of system

파일 모듈은 문서파일을 관리하기 위한 것으로 파일 로드, 저장, 변환 파일 저장 등의 작업을 처리한다. 트리 View 모듈은 변환할 문서 구조를 쉽게 파악할 수 있도록 트리 구조로 나타낸다. Editing 모듈은 원문을 편집하기 위한 도구로서 노드의 추가, 삭제 및 편집하는 기능을 지원한다. Converter 모듈은 원문을 목적 파일로 변환한다. Printing 모듈은 변환된 문서를 PostScript 인터프린터가 내장된 프린팅 디바이스로 출력하는 모듈이다. 에러 처리 모듈은 변환기에서 발생한 예외를 간단한 메시지로 알려주는 기능을 지원한다.

IV. 시스템 설계

4.1 시스템 운영 설계

XSL-FO를 효과적으로 분석, 사용하기 위해서는 XSL-FO가 엘리먼트 단위로 이루어져 있다는 것을 알아야 한다. 즉, 엘리먼트마다 고유의 역할이 있다는 것이다. 본 시스템에서는 이러한 원소들을 효과적으로 처리하기 위해서 엘리먼트 단위로 변환하였다.

1) layout과 content관련 엘리먼트를 구별하고, 엘리먼트 내에서 사용되는 엔티티에 관련된 메서드를 정의한다.
2) 특정 엘리먼트가 없을 경우를 대비해 일반적으로 사용되는 값을 디폴트로 선언한다.
3) XSL-FO에서는 여러 종류의 단위가 사용되기 때문에 단위환산과 같은 몇 가지 프로시저를 생성해서 사용한다.

4.2 변환 프로세싱

컨버터 처리과정은 트리 탐색 순으로 이루어지기 때문에 문서의 상단에 위치하는 엘리먼트에 대한 처리를 하고, 각종 엘리먼트의 엔티티에 대한 변환 과정을 거치게 된다. 다음과 같이 3단계로 구분한다.

첫 번째 단계에서 layout과 관련된 엘리먼트에 주어진 값으로부터 문서를 처리하는 layout 정보를 가져온다. 두 번째에서 텍스트 위주의 처리 과정이 진행된다.

세 번째 단계에서는 입력된 문서가 적용시키기 부적절한 문서이거나, 변환되어야할 엘리먼트나 그 속성이 적절치 못할 경우 또는 PostScript에서 적용될 수 없는 엘리먼트일 경우 '예외'를 처리하여 확인 가능하게 함으로써 프로그램적인 부분을 잘 알지 못하는 사용자들도 원활히 사용하도록 하였다.

처리된 문서는 인쇄기능을 통해 출력결과를 확인할 수 있고, 사용자의 의도에 따라 직접 편집도 가능하다. 그러나 XSL-FO와 PostScript 간의 일 대 일의 완전한 변환이 불가능 하다. 이를 해결하기 위해서 PostScript의 프로시저를 생성해 사용하였다. 예를 들면 문서의 가운데 정렬과 같은 엔터티를 만나게 되었을 때, 텍스트의 길이를 계산하고 문서의 중앙을 찾아 두 부분으로 나눈 뒤 적절한 배치를 하는 프로시저를 생성해야 한다.

4.3 프로시저 자동사용 알고리즘

컨버터 처리를 하기 전에 layout과 관련된 특정 엘리먼트가 문서 내에 존재하는 가를 확인하고, 있지 않을 경우 미리 정의해둔 프로시저를 호출해서

사용한다.(그림 3.. 일반적으로 여러 문서를 살펴보면 XSL-FO에서 사용되는 layout의 단위가 하나의 통일된 단위로 사용되지 않고 있다. 예를 들면 PostScript에서 사용되는 단위 unit으로는 pt(point)를 사용하는데, XSL-FO문서에는 'cm', 'mm', 'inch' 등 다양하게 사용하기 때문에 변환할 때 단위관련 프로시저를 호출하여 사용 하도록 하였다.(그림 4.) 또한 margin값을 할당해 주는 방식이 한 가지로 통일되어 있지 않기에 margin값 선언의 여러 경우를 받아 들여서 하나의 통일된 PostScript 선언으로 바꾸어 주어야 한다.

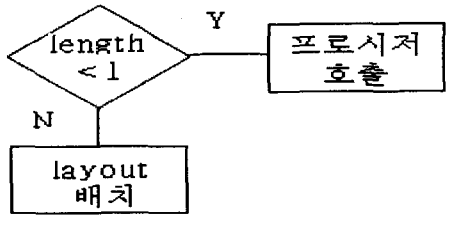


그림 3. 특정 엘리먼트 유무 확인 프로시저
fig .3 Procedure for existence check of specific element

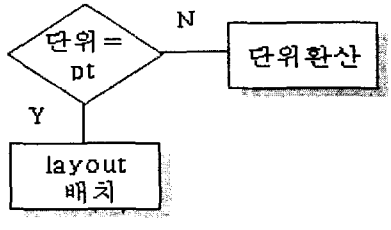


그림 4. 단위 단일화 프로시저
Fig. 4 Procedure for unit unification

Content배치는 적당한 시작점으로 이동해 그곳에서부터 content를 표시하는 것인데, PostScript는 page 넓이 보다 내용이 길더라도 새로운 line으로 이동하지 않는다. 현재 위치(current point)가 page의 끝에서 margin을 뺀 위치좌표가 클 경우 새로운 line에서 남아있는 내용들을 시작하게 하는 프로시저를 생성해야 한다.(그림 4.) 마지막 단락이 끊어져서 인쇄될 경우 다음 페이지로 이동해서 단락을 처리해야 한다. 단락영역 설정 알고리즘은 현재 남아있는 페이지의 공간과 페이지에 기록될 단락이 차지할 영역을 비교해서 단락의 위치를 결정하는 방식이다.

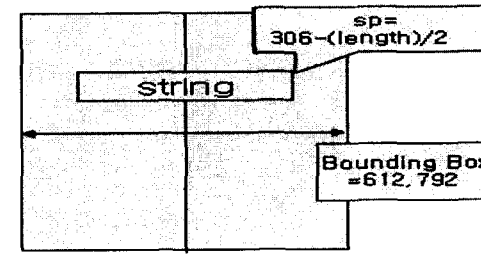


그림 5. 텍스트 정렬 프로시저
Fig. 5 Procedure for text alignment

4.4 PostScript 파일 작성

본 논문에서는 XSL-FO의 오브젝트와 프로퍼티에 따라 PostScript 파일로 변환한다. <layout-master-set>엘리먼트에 포함된 속성과 하위 엘리먼트가 레이아웃 양식을 제어한다. 다음과 같은 엘리먼트와 그의 속성과의 관계를 규정해 놓고 그들의 값들로 실제 layout을 정하게 된다.

paragraph 사이 간격은 <fo:block>엘리먼트의 속성중 'space-after.optimum' 의 값으로 계산되고, page number도 page의 하단 부에 배치시켜야 한다. 이외에도 paragraph시작 부위의 들여 쓰기를 추가할 수 있는데 이때도 PostScript의 좌표계에 대응되는 값을 추가해야 한다.

또한 contents의 내용이 가변적이기 때문에 다음 글의 위치를 동적으로 지정해 주어야 한다는 것이다. 이는 fo가 block단위로 contents를 지정해 주기 때문이다.

PostScript에서 사용되는 Unit는 'pt'(point)를 사용하는데 1point는 1/72 inch이고. 따라서 72point는 1inch이다. 예를 들면 8.5×11inch page는 612×792point넓이가 된다. 또한 이 page의 center는 306,396이 된다. layout될 전체 page의 size나 margin 등의 정보를 받아 PostScript상에 배치 시켜 줘야 한다.[12]

Pagelayout과 관련된 엘리먼트들은 layout-master-set, simple-page-master, page-sequence-master가 있고, layout-master-set는 하나 이상의 simple-page-master를 가지고 있는데, 실제로 page layout을 지정하는 부분이다. 한 페이지에서 region과 margin의 관계를 정의해야 하고, 각 region들은 서로간의 영역들을 겹치지 않게 적당한 간격을 두어야 한다. 하나 이상의 마스터 페이지를 필요로 할 경우 page-sequence-master를 이용해서 간단하게 지정한다.

layout의 중요한 속성 두 가지는 margin과

region 두 가지이다. 이 두 가지 속성으로 레이아웃이 정해지는데, simple-page-master의 page margin과 각 region의 margin 속성의 값이 사용된다. 문서의 시작 부분은 페이지 마스의 왼쪽 margin과 region-start의 값과 region-body의 margin의 합해진 길이만큼 이동한 곳 이된다. 페이지의 상단 부분도 page margin, body-top 등의 길이를 계산해서 이동한 후에 결정된다. 이들이 처리되는 부분들을 컨버터 처리 시작 전에 미리 정의해서 사용하도록 하였다.

Text를 중앙으로 배열을 하기 위해서 먼저 해야 할 것은 PostScript의 X coordination 이 display될 시작점을 찾아야 한다. PostScript는 일반적으로 BoundingBox(default image area)로 612×792를 사용하고 있다. page의 중앙을 찾은 후, text의 길이를 얻어 오는데 이것은 text가 중앙을 기점으로 해서 좌우로 같은 길이만큼 배치된다는 점을 이용한 것이다.

DOM 파서가 XSL-FO 문서를 파싱하면서 원문의 상태를 점검하고 이상이 발견되면 예외를 발생시킨다. 발생된 예외는 사용자가 보기 쉽게 간단한 인터페이스를 통해 전달되고 이로 인해 사용자는 어느 부분에서 잘못되었는지 알 수 있게 된다. 위의 그림처럼 예외 처리 테이블을 미리 만들어 놓고 파싱을 하면서 예외 처리 대상과 비교하게 된다.

4.5 Text Processing

문서의 품질을 높이기 위해서 Text를 변형 시키거나 문서에 이미지를 삽입 하도록 한다. Text는 font, size, rotate, translate , Circular Text 등의 변형을 하도록 한다. 간단 또는 복잡한 이미지를 추가시켜 문서의 출력 품질을 향상 시킬 수도 있다.

V. 시스템 구현

5.1 파일 관리 모듈

이 모듈은 문서파일을 관리하기 위한 것으로 기존의 문서를 로드 하여 문서의 내용에 추가 및 수정이 가능하게 하고, 변환된 파일을 저장하는 기능들을 포함한다.

5.2 문서 타입 검출기

DOM 파서는 XML 문서를 자바 객체 모델로 변환시키는 자바 프로그램 API이다. 한 번 파싱된 문서는 자바 가상 머신의 메모리에 존재하기 때문에 프로그래머는 파일을 직접 접근하지 않고 메모리에 있는 자바 객체를 통해서 문서의 내용을 접근하거나 변경할 수 있다.

본 논문에서 사용하고 있는 문서 타입 검출기는 DOM트리를 순회하면서 문서의 내용을 자유롭게 접근, 출력할 수 있다. 파서를 이용해서 문서를 파싱하고, 파싱된 문서의 가장 상위 노드인 Document노드를 얻어 노드 타입을 알아본다. Node는 노드 타입을 표현하기 위해서 정적 멤버 필드들을 가지고 있다. 이 멤버 필드들은 정수 값으로 표현되고, 다음 표는 멤버필드의 이름과 값을 보여준다.

5.3 트리 View모듈

XSL-FO는 XML 형식을 사용하고 있고, 문서의 내용(엘리먼트, 속성, 속성값, txt)들을 텍스트 창에서 보는 것보다 tree형태로 보게 되면 문서의 구조를 쉽게 파악할 수 있기 때문에 Viewer를 제공해서 문서의 형식을 잘 알지 못하는 사용자도 쉽게 이해할 수 있도록 인터페이스를 제공하였다.

5.4 Editing module

사용자의 효율적인 사용을 위해 원문 수정이 필요한 경우가 발생할 수 있기 때문에 에디터를 추가하여 원문을 수정할 수 있도록 인터페이스를 제공하였다. 이 편집기는 왼쪽에 XSL-FO문서 구조를 트리 형식으로 보여주고, 오른쪽에는 텍스트의 데이터를 입력하도록 되어있다.

에디터 왼쪽의 트리 구조에서 마우스의 오른쪽 버튼을 이용해 자식 노드를 추가, 삭제할 수 있게 하였고, 오른쪽 창에 텍스트 데이터를 입력할 수 있어서 내용 편집이 가능하다.

W3C에서 XML문서 처리를 위해 XML파싱의 표준으로 선택한 것은 DOM(Document Object Model)이다. 트리 형태로 데이터를 저장하는 DOM은 구조적이고 프로그래밍이 용이하다는 장점이 있지만, XML데이터를 통째로 읽어서 트리 구조로 저장한 후에 사용해야 하기 때문에 비효율적이고 느리며, 많은 자원을 소모한다는 결정적인 단점이 있다. 이런 어려움을 극복하기 위해 XML 개발자들은 XML-DEV메일링 리스트에서 지속적으로 연구했고, 새로운 XML문서의 파싱 방법으로 메모리를 적게 차지하면서도 빠르고 효율적인 기술인 SAX(Simple API for XML)를 개발했다. 본 논문의 에디터에는 SAX를 사용해 작성하였다.

이 모듈은 원문을 수정 및 편집하기 위한 기능으로, 트리형태로 나타나는 원문을 간단한 인터페이스를 통해서 확인할 수 있는 것이 특징이다. 특정 노드를 추가, 삭제할 수 있게 하였고, 텍스트 편

집하는 기능을 추가하여 문서의 내용을 변경할 수 있도록 하였다.

5.5 Conversion module

파싱된 문서는 각 type별로 해당 변환을 실행한다. 원문은 block 엘리먼트를 중심으로 내용이 기술되기 때문에 block 엘리먼트의 속성을 중심으로 변환 하였고, 필요한 속성이나 segment가 없는 경우 일반적으로 사용되는 값을 첨부하였다.

5.6 Printing module

다음 그림에서 볼 수 있듯이 변환된 파일을 출력할 수 있는 기능을 제공하였고, 속성을 선택할 수 있는 메뉴를 제공해서 사용자가 원하는 형태로 출력을 얻을 수 있도록 하였다. 원하는 인쇄서비스를 선택해서 페이지를 설정하고 인쇄될 페이지의 모양도 사용자가 원하는 대로 선택할 수 있도록 구현하였다.

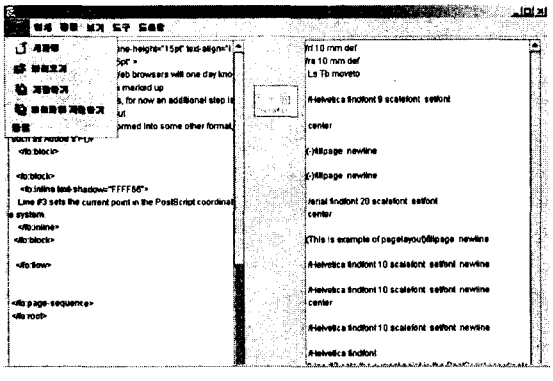


그림 6. 변환기 인터페이스
Fig. 6 User interface

그림 6.은 변환기를 구현한 시스템 화면이다. 그림 2.의 시스템 구성에 따라 파일, 편집, 보기, (변환)도구 그리고 도움말을 주 메뉴로 제공하고 각각은 관련 부 메뉴를 구성하거나 다이얼로그 상자를 통해 변환작업을 제어할 수 있도록 하였다. 아래 그림은 불러들인 XML 문서에 변환기능(도구 메뉴를 사용)을 적용한 예이다. 좌측에 XML 문서를 보이고 우측에는 변환된 PostScript 파일을 보인다. 그림 7. 은 PostScript 인터프리터가 내장된 프리터로 출력한 예이다.

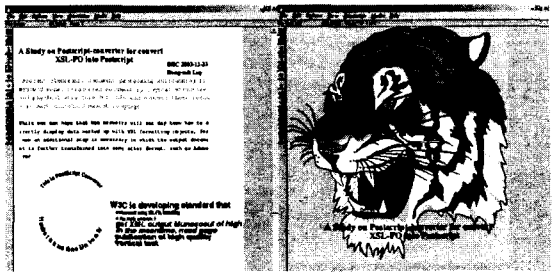


그림 7. 변환된 문서
Fig. 7 Example of the converted document

VI. 결 론

본 논문에서는 웹 문서에서 생겨나는 인쇄물에 대한 만족스럽지 못한 결과를 해결하기 위해 XSL-FO에 대한 PostScript로의 변환에 대한 시스템 설계에 관하여 기술 하였다. XSL-FO문서를 트리 형태의 노드라는 요소로 구성되는 DOM API를 사용하여 논리 구조와 레이아웃 구조로 구조화하고, 이들을 엘리먼트와 프로퍼티를 중심으로 PostScript의 처리 명령어로 변환하는 과정을 연구하였다.

본 논문에서 제안한 프로시저 자동할당 알고리즘에 의해 할당된 프로시저로 PostScript에서 지원할 수 없는 부분을 처리하도록 하였고, 파일 변환할 때 필요한 부분을 미리 정의해 둠으로써 불필요한 처리과정을 줄일 수 있었다.

현재 페이지 기술언어로 사용되고 있는 PostScript 인터프린터가 내장된 프린터를 출력장치로 하여 변환된 PostScript 파일을 출력한 결과, 고품질의 문서를 얻을 수 있었다. 따라서 XML-FO를 PostScript에 적용함으로써 최적의 인쇄물을 얻을 수 있음을 확인하였다. 그러나 본 변환기는 전처리 과정에서 DOM/SAX 파서를 필요로 하기 때문에 현재로서는 변환기를 브라우저에 plug-in 으로 등록한다 하여도 직접적인 변환이 불가능한 문제점이 있다. 따라서 향후 파싱기능을 포함하도록 변환기를 확장한다면 브라우저 상에서 직접 PostScript로 변환된 고품질의 출력문서를 얻을 수 있을 것이다.

참고문헌

[1] 김차종, 동적포맷팅 방식을 이용한 X윈도우

/Motif상의 전자출판 시스템에 관한 연구, 1990. 12
[2] Jon Barrett, Kirk Reistroffer, Designing a Raster Image Processor, BYTE, pp.171-180, May, 1987.
[3] Denis G. Pelli, Programming in PostScript ,BYTE, pp185-202, May,1987
[4] John Seybold/ Fritz Dressler by, Publishing From The DESKTOP, A Bantam Book, 1987
[5] Michael Sweeney, Electronic Publishing, UNIX/WORLD, pp.28-45, April, 1986
[6] http://xml.apache.org
[7] http://www.AntennaHouse.com
[8] http://www.lunasil.com
[9] 김춘웅/ 이명진 by, 자바 개발자를 위한 XML 프로그래밍, 한빛 멀티미디어, 2002
[10] KurtCale by, PROFESSIONAL XSL, wrox, 2001
[11] http://www.w3c.org
[12] Ed Taft/ Jeff Walden by, PostScript Language Reference Manual, Addison Wesley, 1997
[13] Dr.MichaelB.Spring & DavidDubinby, Hands-On PostScript, HAYDE

저자소개

유동석(Dong-Sok Luy)

한밭대학교 컴퓨터공학과
('02 공학사)
한밭대학교 검퓨터공학과
('04 공학석사)
※관심분야 : XML, Web Service, 인터넷 프로그래밍

김차종(Cha-Jong Kim)

광운대학교 컴퓨터공학과
('84공학사)
광운대학교 컴퓨터공학과
('86 공학석사, '91 공학박사)
Pittsburgh of University, U.S.A 초빙교수('97.7-'98.7)
국립한밭대학교 정보통신·컴퓨터공학부 교수
※관심분야 : XML 기반 Web Service, 컴퓨터그래픽스 Sementic Web, Mobile Programming