한국산학기술학회논문지
Vol. 11, No. 11 pp. 4583-4591, 2010

동적 편집과 포맷팅 기능을 갖는 XML 기반의 가변 데이터 출판 시스템

임광택[1*]
[1]호원대학교 컴퓨터게임학부

XML-based Variable Data Publishing System with Dynamic Editing and Formatting Function

Kwang-Taeg Lim[1]
[1]**Division of Computer & Game, Howon University**

요 약 사용자가 직접 템플릿 규칙을 코딩하고 편집해야하는 기존의 XML 기반 가변데이터 출판 환경은 일반 사용자에게 문서제작에 많은 부담을 주고 어려움을 갖게 한다. 또한 매뉴얼이나 기술문서와 같은 대용량의 가변 XML 문서 처리시 편집을 위한 빠른 응답속도를 제공하기 위하여 신속한 포맷팅이 제공되어야 하지만 기존의 일괄적인 처리 방식으로는 해결하기 어렵다. 본 논문에서는 템플릿 기반의 가변문서의 처리 결과를 WYSIWYG 화면상에 표시하여 대화식 방식을 통해 템플릿을 편집할 수 있도록 하며 대용량 문서에 대해서도 사용자의 요청에 따라 신속하게 포맷팅하는, 동적 편집과 포맷팅 방식의 기능을 갖는 가변데이터 출판 시스템을 제안한다. 제안된 시스템은 개인이나 기업 또는 지역 등과 같이 개별적인 특성에 따라 변동되는 다량의 가변 데이터를 가지는 맞춤형 문서 제작을 위해 효과적으로 사용될 수 있으며, 입력문서와 템플릿 문서, 포맷된 결과문서는 모두 W3C에서 제안하는 XML, XSLT, XPath의 표준을 수용함으로써 웹 문서처리 시스템으로도 쉽게 확장될 수 있도록 하였다.

Abstract Existing XML-based variable data publishing, in which a user has to manually prepare and edit template rules, is rather difficult for general users to create documents. Especially when processing large variable XML documents such as manuals or technical documents, fast document formatting is required to provide fast response speed for editing, which the existing batch processing cannot provide. This paper proposes a variable data publishing system with dynamic editing and formatting function, which support fast formatting upon user's request for large volume documents as well as for template editing through interaction by displaying the result of template-based variable documents on WYSIWYG screen. Proposed system can be effectively used for creating customized documents with many variable data that can be changed according to individual characteristics such as individual, company or area; source documents, template documents and formatted documents adopt XML, XSLT and XPath standards suggested by W3C, which facilitates extension to web document processing system.

Key Words : XML, XSLT, Document Template, Dynamic Formatting, Electronic Publishing

1. 서론

최근 디지털 인쇄 및 문서처리 기술의 발전에 따라 온라인/오프라인에서 맞춤형 문서의 제작 과 인쇄, 주문 등의 기능을 편리하게 이용할 수 있는 자동화된 출판에 대한 관심이 고조되고 있다[1]. 특히 개인 및 기업, 지역, 학교 등과 같이 개별적인 특성으로 인하여 수시로 변동되거나 다른 상태 값을 가지는 가변 데이터 출판(Variable Data Publishing)에 대한 요구가 급증하고 있다. 이러한 출판 방식은 기존의 레이아웃 기반의 WYSIWYG 전자

본 논문은 2010년 호원대학교 교내학술연구비 지원에 의하여 연구되었음.
*교신저자 : 임광택(ktlim@howon.ac.kr)
접수일 10년 10월 15일 수정일 10년 10월 25일 게재확정일 10년 11월 19일

출판방식과는 다르게 내용기반의 출판을 가능하게 하는 새로운 전자 출판 분야로 주목 받고 있다.

가변데이터 출판은 문서내용에 대한 재활용성을 중요시하므로 내용과 스타일, 레이아웃을 분리하여 처리하는 것이 일반적이다[2,3]. 이를 위해 현재 대부분의 가변문서 처리 시스템은 템플릿(template)을 기반으로 문서내용을 처리하는 방식을 채택하고 있다.

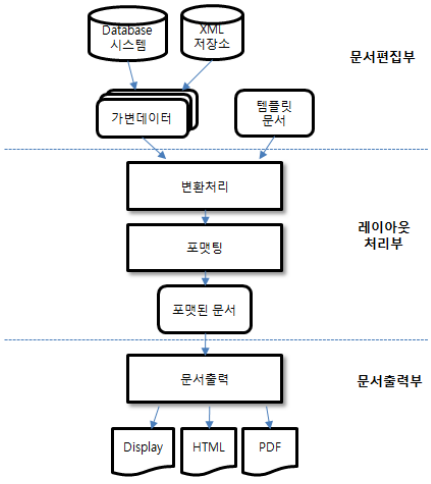
그러나 이러한 규칙기반의 템플릿 처리 방식은 장점과 단점을 가지고 있다. 템플릿은 가변 문서에 적용하여 일괄적으로 빠르게 처리할 수 있는 장점이 있지만, 반면에 프로그램적인 방식으로 편집되기 때문에 최종 결과를 즉시 알기가 어렵고 템플릿 규칙에 대한 상당한 지식이 요구된다. 따라서 템플릿의 작성 및 편집 작업은 전문가로 국한되며, 문서의 양이 많거나 편집이 자주 일어나는 경우에는 매우 비효율적이다. 현재 이러한 문제를 개선하고 해결하기 위한 가변 데이터 출판에 관한 저작도구 및 처리 방식에 관한 다양한 연구가 진행되고 있다.

이에 본 논문에서는 이러한 문제를 해결하기 위하여 동적 편집과 포맷팅 처리 방식을 갖는 가변데이터 출판 시스템을 제안한다. 동적 편집 방식이란 기존의 가변문서 처리환경과는 다르게 WYSIWYG 화면상에서 대화형 다이얼로그 등을 통해 소스 문서와 템플릿을 수정하고 그 결과를 확인할 수 있는 편집환경을 의미한다. 동적 포맷팅은 사용자의 요청에 따라 선택된 페이지별로 신속하게 포맷팅 처리하여 편집 응답속도를 제공하는 포맷팅 방식이다. 제안된 시스템은 XML 기반의 표준 언어를 이용하여 입력문서, 템플릿, 포맷된 문서를 정의하고 처리하도록 구현되어졌으며, 웹 기반의 문서처리 시스템으로도 쉽게 확장될 수 있는 구조를 갖는다.

2. 관련연구

2.1 가변데이터 문서처리의 개요

가변문서를 처리하는 시스템은 그림 1과 같이 크게 세 가지로 구분되는 구성요소를 가진다. 즉, 문서 편집부, 레이아웃 처리부, 문서 출력부로 구성된다. 문서 편집부에서는 데이터베이스 또는 저장 장치로부터 가변데이터를 추출하여 소스 문서를 구성하며, 또한 소스 문서의 구조를 변환하고 레이아웃을 처리하기 위한 템플릿(template) 문서를 작성한다. 그리고 이러한 입력 문서는 템플릿을 적용하여 결과 트리로 변환하는 처리 과정을 거친다. 레이아웃 처리부에서는 텍스트, 이미지 등의 레이아웃 처리를 수행하여 최종 결과 문서를 생성한다. 이 과정에서 생성된 포맷된 문서는 다음 단계인 문서 출력부로 전달되어 브라우저 등 화면에 표시되거나 프린터 출력을 위한 포맷 형식으로 변환되어진다.



[그림 1] 가변데이터 문서처리 시스템

2.2 XML 출판 관련기술

가변데이터 문서의 장점은 논리적 구조만으로 구성된 가변 데이터들에 레이아웃 및 스타일을 적용하여 다양한 출력 형식의 결과를 얻는 것이다. XML은 이러한 접근방법을 가장 잘 수용하고 있는 표준 마크업언어 중의 하나이며, 출판을 위한 관련 표준들을 제시하고 있다.

XML은 문서의 내용과 표현을 분리하고 있다. 따라서 XML 문서는 데이터의 구조와 의미에 관한 정보만을 기술한다. XML에서 문서의 레이아웃에 대한 실질적인 표현은 XSL (eXtensible Stylesheet Language) 등과 같은 스타일시트언어를 사용한다.

XSL은 XSLT, XPath, XSL-FO 세가지 표준으로 구성되어 있다. XSLT[4]는 XML 문서를 다른 형태의 문서로 변환하거나 재구성하기 위한 XML 기반의 변환 언어이다. XSLT 스타일시트는 XML 문서를 변환하기 위한 템플릿 규칙들로 구성되며, XSLT 처리기를 통해 처리되어 결과 문서를 생성한다. XPath[5]는 XML 문서의 구조에 접근하기 위한 언어이다. 문서구조의 노드를 찾기 위해 패턴 식(pattern expression)을 사용하며, 데이터 타입과 연산자, 함수 등을 함께 사용하여 패턴식을 만들 수 있다. XSL-FO [6]는 XML의 논리적 요소에 포맷팅 객체를 지정하여 다양한 매체에 적용되는 출력결과를 생성할 수 있도록 하는 스타일언어이다. 이를 통해 문서를 재구성하거나 여과(filter)하여 책, 신문, 잡지 등과 같은 복잡한 레이아웃 문서를 생성할 수 있다.

2.3 전자출판시스템의 특성 분석

문서처리방식은 일괄처리 방식과 대화형 처리 방식으로 구분 할 수 있으며, 처리방식에 있어 각각 장단점을 가진다. 일괄처리 방식은 편집과정이 불편한 반면, 대량의 문서를 정해진 규칙에 따라 빠르게 처리하여 결과를 생성할 수 있는 장점을 가진다. 그에 비해서 WYSIWYG 방식은 최종결과의 화면 위에서 편집할 수 있는 장점이 있지만 편집을 위한 내부의 많은 상태정보와 포맷된 결과를 가지게 되므로 대량의 문서를 처리할 경우 시스템 성능이 저하되는 주요 요인이 된다. 또한 문서 내용과 스타일정보, 레이아웃 정보들이 문서에 강하게 연결되어 있기 때문에 다른 크기를 갖는 매체로 문서를 재배치(reflow)하는 것이 매우 어렵게 된다.

2.4 가변데이터 시스템의 분석

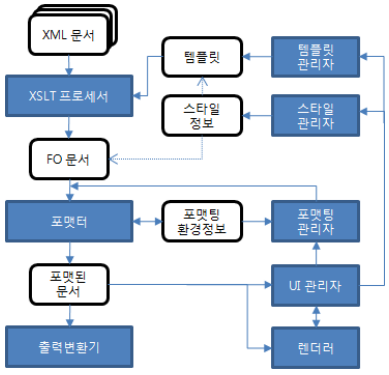
가변데이터 문서를 위한 문서 포맷, 고속인쇄, 구현기술에 대한 여러 가지 연구 및 개발이 진행되어 왔다. UCancode[7]는 카피 홀 모델에서 데이터베이스 필드를 연결하여 가변문서를 인쇄할 수 있도록 페이지 기반의 편집 기능을 제공하고 있다. Quint[8]는 Amaya 편집기 내에서 XML 문서의 구조화된 편집 방법에 대해서 연구를 수행하였고, 결과 문서에서 특정 엘리먼트에 대한 어떠한 스타일이 변경되었는지를 알기 위해 CSS 스트일시트를 분석하여 처리하는 편집방법을 제시하였다. Villard[9]는 실행상태의 정보와 연관된 패턴을 가지고 소스 또는 변환된 것으로부터 변경될 필요가 있는 출력부분만 재계산하도록 하는 점진적 XSLT 처리 방식을 제안하였다. 이와 유사한 연구로 명령단위가 아닌 엔티티와 결과 문서의 파일단위로 재실행해서 크기가 큰 대량의 XSLT 문서를 점진적인 방법으로 처리하는 연구가 이루어졌다[10]. 또한 XML 문서의 저장 및 검색을 위한 관련 연구가 진행되어 왔다. DBMS 종류와 플랫폼에 독립적인 XML 문서의 변경 탐지 기능을 갖는 통합 리파지토리 시스템이 제안되었고[11], 구조기반 인덱싱 모델을 통해 XML 문서의 효율적인 구조검색을 위한 연구가 이루어졌다[12]. 이외에도 HP Exstream, Pageflex와 같은 상용화된 제품에서는 서버기반의 맞춤형 출판 방식과 웹 기반의 온라인 템플릿 편집과 자동화된 출판 방식에 대한 연구가 이루어져왔다.

3. 가변데이터 출판 시스템 설계

3.1 시스템의 전체 구성

본 논문에서 제안한 가변 데이터 출판 시스템은 크게 XSLT/XPath 처리기, 템플릿 및 스타일 관리자, 포맷터, 포맷팅 관리자, UI 관리자, 렌더러, 출력 변환기로 구성되어 있다. 그림 2는 시스템을 구성하는 각 요소와 실제 가변문서 처리를 위한 요소들 간의 상호 관계 및 전체 흐름을 보여주고 있다.

XML 문서와 템플릿, 레이아웃 정보는 XSLT 프로세서를 통해 FO(Fomatting Object) 문서로 변환된다. 포맷터는 FO 문서를 처리하여 포맷된 문서와 포맷팅 환경정보를 생성한다. 포맷된 최종 결과는 렌더러를 통해 화면에 표시되거나, 출력 변환기를 통해 PDF, HTML 등과 같은 출력 파일로 생성되어진다. 포맷팅 환경정보는 동적 포맷팅을 위해 포맷팅 관리자로 전달된다. UI 관리자는 사용자 상호작용을 제어하며 포맷터와 템플릿, 스타일 관리자를 호출하는 기능을 담당한다. 템플릿과 레이아웃 관리자는 화면에 표시된 결과문서 위에서 템플릿과 레이아웃을 수정할 수 있도록 대화형 편집 다이어로그 등의 사용자 인터페이스 기능을 제공한다.



[그림 2] 전체 시스템 구성도

3.2 동적 편집기

3.2.1 편집 문서의 구성과 처리

본 논문에서 가변데이터 문서는 크게 XML 데이터, 템플릿, 스타일 정보 세 개의 파일로 구성된다. 즉, 편집을 위한 문서는 데이터, 구조, 표현으로 분리하여 관리한다. 이러한 가변문서의 구조는 확장성과 융통성에서 많은 장점을 제공하며, 특히 동적편집을 위한 데이터 관리와 매핑을 위한 처리에서 많은 유용함을 준다.

가변데이터는 XML 편집도구 또는 DB로부터 추출되어 XML 문서로 구성된다. 내장된 XML 파서는 XML 문서를 파싱하여 문서의 유효성을 검증하고 DOM 객체 트리를 생성한다. 이 객체트리는 템플릿의 노드를 지정하기

위한 XPath 식을 만들기 위해서 템플릿 관리자에서 이용된다.

템플릿과 스타일 파일은 XSLT와 XPath 언어를 사용하여 표현되며, XSLT 프로세서를 통해 처리된다. 템플릿은 크게 두 부분으로 구성된다. 한 부분은 XML 문서의 노드를 지정하는 경로식이며, 다른 부분은 결과 트리로 변환될 규칙들로 구성된다. 이를 위해 사용자가 직접 규칙을 입력하는 방식이 아닌 대화형 편집기능을 갖는 템플릿 관리자를 제공한다. 스타일 파일은 문서의 레이아웃을 지정하기 위한 페이지, 블록, 인라인, 테이블, 리스트, 이미지 등과 같은 클래스로 분류되며 각각에 대한 포맷팅 객체 속성으로 구성된다.

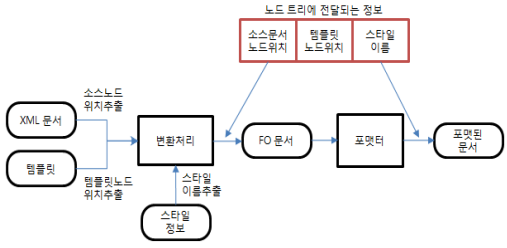
XML 데이터와 템플릿, 스타일 파일은 다음 처리 단계인 포맷터의 입력으로 사용하기 위해 XSLT 프로세서를 통해 결과문서 즉, FO 객체 트리로 변환된다. FO 객체 트리는 입력문서를 템플릿과 스타일 속성을 적용하여 재구성되는 XML 문서이다. 입력문서가 FO 객체트리로 변환된 후 이 두 개의 문서사이에는 연관관계를 갖지 않지 않으므로, FO 객체트리를 통해서 입력문서를 만드는 역변환은 불가능하다. 하지만 FO 객체트리를 생성할 때 어느 템플릿을 통해서 생성되어졌는지 원인에 대한 소스 위치를 제공할 수 있다. 이를 통해 동적편집을 위한 역추적이 가능하며 템플릿 관리자와 스타일 관리자를 통해 해당 속성을 변경할 수 있다.

3.2.2 역추적을 위한 매핑 정보

앞서 기술한데로 템플릿을 통해 변환된 FO 객체트리는 입력문서와 연관성을 갖지 않으므로, 동적 편집을 위해서는 매핑 정보를 가질 필요가 있다. 이 매핑정보는 실제로 포맷된 최종 결과문서 위에서 사용자 상호작용에 따라 선택된 텍스트 또는 이미지, 그룹 및 플로우 속성 등을 변경하기 위해서 사용된다.

이와 같은 동적편집을 위해서 다음 두 가지의 전제 조건에 대한 해결이 필요하다. 하나는 소스 템플릿 문서의 어느 부분에서 편집이 일어나는지 정확한 위치를 알아야 하고, 다른 하나는 결과 엘리먼트에 대해 어떠한 편집속성이 적용되는지를 식별할 수 있어야 한다.

이를 해결하기 위해서는 결과문서의 각 엘리먼트들은 소스 템플릿에 대한 위치 정보와 편집속성에 대한 정보를 가지고 있어야 한다. 본 시스템에서는 변환과 포맷팅의 각 처리 단계에서 이들 정보를 전달하는 구조로 설계하였다. 그림 3은 매핑을 위한 처리 과정을 나타내고 있다.



[그림 3] 매핑을 위한 처리 과정

소스 문서의 위치는 문서 파싱 단계에서 생성된 DOM 트리 객체로부터 만들어지는 노드의 깊이 정보이며, 템플릿 노드의 위치는 템플릿 관리자를 통해서 사용자가 각 템플릿에 부여하는 이름 정보이고, 스타일 이름은 템플릿 관리자에서 템플릿 규칙을 만들 때 적용하는 스타일에 대한 이름정보를 나타낸다.

3.2.3 템플릿 문서 구조

템플릿 관리자의 주요 기능은 XSLT 형식의 template 구조를 만드는 것이다. XSLT의 template을 만들기 위해 대화형 다이얼로그 인터페이스 기능이 제공되며, 패턴식 뿐만아니라 페이지 레이아웃, 스타일 정보들을 template에 연결하는 기능을 갖는다.

전형적인 XSLT template은 다음과 같이 매치 패턴과 규칙으로 구성된다.

```
<xsl:template match="pattern">
        [template 규칙]
</xsl:template>
```

본 시스템에서는 동적편집을 위한 매핑정보를 삽입하기 위하여 template 노드의 구조를 그림 4와 같이 확장하였다.

```
<xsl:template match="pattern" vdp:name="" vdp:class="" vdp:style="">
    <vdp:element vdp:name="">
        <xsl:apply-templates [select=""]/>
    </vdp:element>
</xsl:template>
```

[그림 4] 템플릿 문서 구조

XML 표준에서 사용하는 이름공간과의 충돌을 피하기 위하여 vdp라는 이름공간을 사용한다. name 속성은 템플릿 자체의 유일한 이름을 나타낸다. class 속성은 포맷터 객체 트리에서 사용되는 클래스 유형을 나타내며, 6가지

유형을 제공한다. 즉, inline, block, image, list, table, link 클래스 유형을 갖는다. style 속성은 template에 적용되는 스타일 속성 이름을 나타낸다. vdp:element는 FO 객체 트리에 생성되는 클래스 유형을 나타내는 엘리먼트 이름이다. xsl:apply-templates은 패턴과 일치하는 입력문서의 노드들을 재귀적으로 처리하기 위한 XSLT 명령이다.

3.2.4 스타일 문서 구조

스타일 정보는 포맷팅 속성을 표현하는 선언적 마크업 구조를 가진다. 표 1에 시스템에서 사용되는 스타일 속성들을 분류하여 표시하였다. 스타일 정보는 포맷팅 객체의 클래스 유형에 따라 다르게 적용된다. 예를 들어, inline과 block 객체는 Font, Tab 그룹이 적용되지만, image 객체에는 적용되지 않는다. 이를 위해 스타일 관리자는 포맷팅 객체의 유형에 따라 포맷팅 속성에 대한 사용자 다이얼로그를 다르게 표시한다.

[표 1] 스타일 정보의 분류

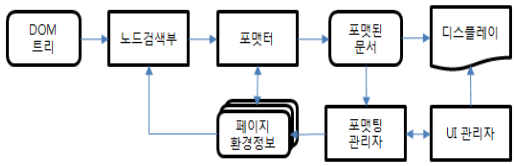
그룹	속성
Font	name, size, color, weight, style
Alignment	left, right, center, justify
Horz.	indent, margin:left,right word space, char. space
Vert.	line space, para space:top, bottom
Break.	page, column, widow, orphan
AutoNum	bullet, number counter, number style, position
Tab.	position, alginment, leader
Border	color, width, style
Misc.	horz./vert. writing, dropped capital, padding

스타일 정보는 템플릿 문서구조에 포함하지 않고 독립적인 형태 즉, 파일로 관리된다. 스타일을 템플릿과 분리하면 포맷팅 속성을 파싱하고 편집을 위한 매핑 처리가 수월한 장점을 가진다. 또한 WYSIWYG 방식의 시스템과 같이 포맷팅 속성을 다이얼로그 형태로 제공하기 위한 처리가 간편해지며, 시스템에서 정의한 스타일 구조를 CSS2와 같은 표준 스타일시트로 쉽게 변환할 수 있게 한다.

3.3 동적 포맷터

3.3.1 처리 모델

본 논문에서 제안한 동적 포맷팅의 기본 처리방식은 사용자의 선택에 따라 점진적으로 포맷팅을 수행하는 것이다. WYSIWYG 화면에서 사용자가 특정 페이지를 선택하면, 해당 페이지를 기준으로 선행하는 페이지를 조사하여 가장 최근까지 변화되지 않은 페이지 이후부터 포맷팅을 수행한다. 그림 5는 동적 포맷팅 방식의 처리 모델을 나타내고 있다.



[그림 5] 동적 포맷팅 방식의 처리 모델

동적 포맷팅의 가장 핵심적인 부분은 페이지 환경정보이다. 페이지 환경정보는 페이지마다의 의존성을 추적할 수 있도록 포맷터가 저장하는 상태 정보이다. 문서를 처음으로 포맷팅하는 경우 페이지 환경정보는 없으나, 이후 포맷팅을 할 때 마다 해당페이지에 대한 페이지 환경정보가 자동으로 저장된다. 사용자가 특정 페이지를 선택하면 UI 관리자를 통해 포맷팅 관리자에게 포맷팅에 대한 요청이 이뤄지고, 포맷팅 관리자는 페이지 환경정보로부터 유효한 페이지인지를 검사한다. 여기서 유효한 페이지란 포맷 처리를 행한 페이지중 문서 레이아웃이 변경되지 않은 페이지를 의미한다.

해당 페이지가 유효한 페이지이면 포맷팅을 하지 않는다. 만약 유효한 페이지가 아니면 포맷팅 환경 정보를 토대로 해당 페이지에 대한 포맷팅을 수행하게 된다. 이때 포맷팅을 위한 노드의 시작위치는 페이지 환경정보로부터 추출되어 그 위치로부터 포맷팅을 수행한다. 동적 포맷터는 이와 같이 페이지의 의존성에 따라 포맷팅의 여부를 결정할 수 있고, 재처리에 따른 계산비용을 크게 줄일 수 있다.

3.3.2 페이지 환경정보

페이지 환경정보는 크게 정적변수, 동적변수로 구분할 수 있다. 정적변수는 페이지 포맷팅을 하기 전에 초기화에 사용되고 포맷팅 도중 변경되지 않는 정보들이다. 동적변수는 사용자의 상호작용으로 인해 포맷팅 도중 변경되는 정보들이다. 이 변수들은 해당 페이지의 포맷팅을 위한 초기화 정보로 사용된다. 표 2는 페이지 환경정보를 분류하여 적용 범위와 포맷팅 속성을 보여주고 있다.

[표 2] 페이지 환경변수의 분류

변수 유형	적용 범위	포맷팅 속성(그룹)
정적 변수	문 서 전 체 및 페이지	page layout
동적 변수	문서전체	auto numbering, TOC, index, footnote, cross-reference
동적 변수	텍스트	font, alignment, horz./vert. attrib, break, tab, border, writing mode, dropped capital.

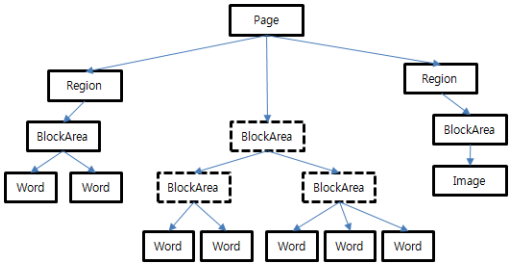
또한 페이지 환경정보는 페이지마다의 포맷팅의 시작 위치를 알기 위해 다음과 같은 정보를 가진다.

- node_position: FO 객체 트리의 노드 위치
- text_position: 텍스트의 시작 위치

node_positon은 페이지 경계에서 마지막으로 처리된 FO 객체트리의 노드 위치이다. 예를 들어, inline, block, list와 같이 포맷팅 객체의 클래스 유형을 나타내는 노드가 이에 해당된다. text_position은 node_position 내에 있는 텍스트로 페이지 경계에서 마지막으로 처리된 텍스트의 위치이다. 이들 두 정보는 다음 페이지의 포맷팅 시작 위치를 가리키기 위해서 사용된다.

3.3.3 포맷된 문서 구조

포맷팅된 결과는 포맷된 문서(fomatted document)로 저장된다. 포맷된 문서는 렌더러를 통해 화면에 표시되거나 출력 변환기를 통해 PDF, HTML와 같은 다른 구조의 문서포맷으로 변환하기 위해 사용된다. 본 시스템에서는 그림 6과 같은 트리 구조를 갖는 포맷된 문서를 XML 언어로 설계하였다.



[그림 6] 포맷된 문서 트리 구조

포맷된 문서의 각 노드 정보는 사각형 영역 모델(rectangular area model)의 개념을 사용 한다. Page는 페이지 노드를 나타내며, Region은 하위 요소의 전체 레이아웃 영역을 나타내는 노드이다. BlockArea 노드는 텍스트와 이미지의 표시 영역이며 내포되어 나타날 수 있다. BlockArea가 내포되는 것은 템플릿 규칙에 의해서 변환된 입력문서의 엘리먼트간의 종속관계를 표현하는 것으로 최종 결과트리에서 원본 문서에 대한 매핑하기 위한 정보로 사용되어진다. Word는 텍스트의 단어 정보를 나타내며, Iamge는 이미지 내용을 표현하기 위한 정보이다.

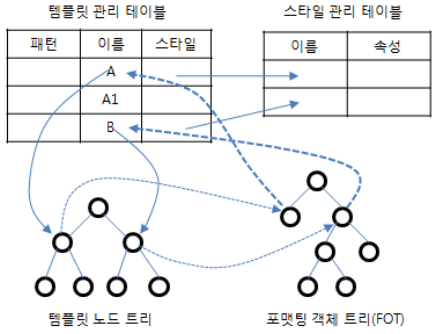
4. 구현 및 고찰

4.1 구현

본 논문에서 개발한 시스템의 구현환경은 Windows Vista 운영체제에서 동작하며, 구현언어는 Microsoft Visual C# 3.0을 사용하였고, XML 파서 및 XSLT 처리기는 Microsoft 사의 COM으로 구성된 MSXML을 사용하였다.

4.1.1 템플릿 관리자 구현

사용자 다이얼로그를 통해 template 규칙을 작성하고 편집하는 템플릿 관리자를 구현하였다. 템플릿 관리자의 중요한 부분은 각각의 템플릿 과 포맷된 문서 사이에 매핑 정보를 유지하도록 하는 것이다. 이를 위해 그림 7과 같은 템플릿 관리 테이블 구조를 이용하였다.



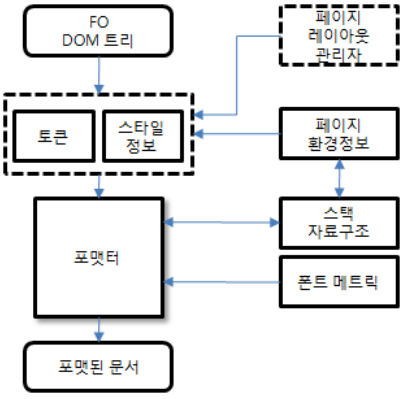
[그림 7] 템플릿 관리를 위한 데이터 구조

템플릿 관리 테이블에서 각 행은 하나의 <xsl:templeate> 엘리먼트에 대한 노드 정보를 나타내며, 이들 정보는 패턴, 템플릿 이름, 스타일 이름들로 구성된다. 패턴은 <xsl:templeate> 엘리먼트의 pattern 식을 나타낸다. 이 패턴식은 사용자가 직접 입력하는 방식이 아닌,

사용자가 소스 문서의 DOM 객체 트리 구조에서 노드를 선택하고 이에 따라 시스템에서 자동으로 패턴식을 입력하도록 구현하였다. 스타일 이름은 템플릿 규칙에 적용되는 스타일 속성 정보에 대한 이름이다. 템플릿 각각을 식별할 수 있도록 템플릿에 유일한 이름을 지정하는 방식을 사용하였다. 이러한 템플릿 이름은 변환 및 포맷팅 과정에서 포맷팅 객체 트리와 포맷된 문서에도 전달되어진다. 따라서 최종적으로 포맷된 가변문서의 인스턴스에서 원본 템플릿의 정확한 위치를 찾을 수 있다. 본 논문에서 동적편집은 이러한 템플릿의 매핑 정보를 토대로 실현되어진다.

4.1.2 포맷터 구현

입력문서의 변환처리를 통해 생성된 FO 결과트리(FOT)의 추상적 의미를 해석하여 실질적인 페이지 구조로 배치하는 포맷팅 처리기를 구현하였다. 또한 선택된 페이지에 따라 의존성을 추적하여 동적으로 포맷팅 하는 기능을 갖도록 포맷터를 구현하였다. 그림 8에 구현된 포맷터의 구조를 보인다.



[그림 8] 포맷터의 구조

포맷터의 처리 방식을 요약하면 다음과 같다. 템플릿을 적용하여 생성된 FO 문서는 포맷팅처리 전 단계에서 DOM 트리로 구성된다. 이 트리는 루트로부터 탐색되어지며, 각 노드 단위로 토큰과 스타일 정보를 추출하여 포맷터의 입력으로 사용한다. 토큰(token)은 트리의 각 노드를 구성하는 엘리먼트 이름으로부터 추출되어지며, 스타일 정보는 템플릿 속성으로 지정된 스타일 이름으로 스타일 관리자로부터 포맷팅 속성을 참조하기 위해 사용되어진다.

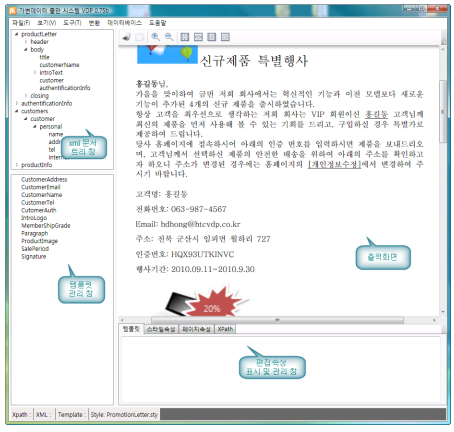
포맷터는 페이지 레이아웃 관리자로부터 페이지 속성 값을 참조하여 페이지 영역, 머리말, 꼬리말, 본문 영역 정보 등을 포맷터의 내부 자료구조인 스택에 설정한다. 또한 포맷터는 페이지의 포맷팅 시작 위치 정보를 페이지 환경 정보로부터 참조하여 스택에 설정한다.

이와 같은 페이지 포맷팅을 위한 초기화 과정이 마쳐지면 포맷터는 실질적인 페이지 레이아웃 처리를 위한 포맷팅 과정을 수행한다. 즉, 텍스트, 이미지, 테이블 등에 대한 배치 작업을 처리한다. 페이지 경계에서 적용되는 inline, block 포맷팅 객체의 속성은 다음 페이지에 직접적인 영향을 미치므로 동적 포맷팅을 위한 다음 페이지의 초기 환경을 구성하는 중요한 정보가 된다. 따라서 이러한 정보는 페이지 환경변수에 저장된다.

텍스트 포맷팅 객체는 하나의 상위 객체가 여러개의 내포된 하위 객체를 포함 할 수 있기 때문에 포맷팅 명령의 유효범위를 효과적으로 관리할 필요가 있다. 위를 위해 스택 자료 구조를 사용하였다. 페이지에 대한 포맷팅 과정을 마치면 포맷터는 앞서 기술한 포맷된 문서 구조로 결과 문서를 저장한다. 이 문서는 다음 처리 단계인 문서출력부로 넘겨져 화면출력을 위해 사용되거나, 프린터 출력을 위한 포맷으로 변환되어진다.

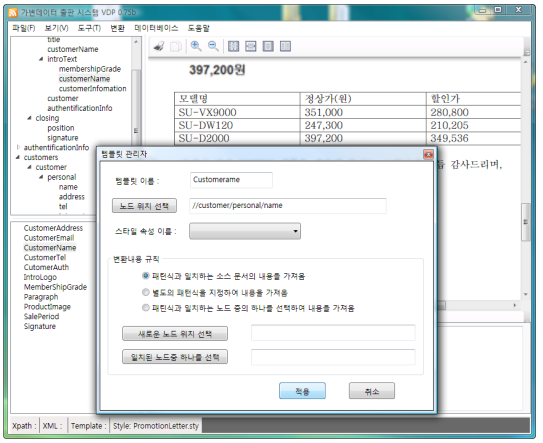
4.1.3 구현 결과

그림 9는 전체시스템의 화면 구성을 보이고 있다. 좌측에 보이는 XML 문서 트리 창은 XML 가변데이터 문서의 구조를 표시하고 노드의 위치를 관리하며, 템플릿 관리자 창은 작성되는 템플릿들의 이름을 표시하고 템플릿 규칙을 편집 할 수 있는 기능을 제공한다. 템플릿을 적용하여 포맷팅 처리 결과는 화면 중앙의 출력화면에 표시된다. 그리고 화면 하단에 있는 편집 속성 창들은 템플릿, 스타일 등의 편집 속성들을 표시하고 관리하는 기능을 갖는다.



[그림 9] 시스템 화면 구성

그림 10은 템플릿 규칙을 설정하는 예를 보이고 있다. 가변데이터의 소스위치에 대한 패턴식은 XML 문서 트리창에 표시된 노드를 선택하면 자동으로 XPath식이 구성되도록 하였으며, 추가적으로 템플릿에 적용될 스타일 속성, 변환 규칙에 관한 속성들도 다이얼로그 화면의 메뉴를 통해서 입력되고 편집될 수 있도록 하였다.



[그림 10] 템플릿 규칙 설정 예

4.2 고찰

본 논문은 템플릿 기반의 가변문서 처리결과를 WYSIWYG 화면상에 표시하여 동적으로 편집하고 대용량 문서에 대해서도 신속한 응답속도를 제공하는 가변데이터 출판 시스템의 설계 및 구현에 관한 것이다. 이를 위해 XML, XSLT, XPath와 같은 W3C 표준 마크업언어들을 사용하여 가변데이터 문서, 템플릿, 스타일 정보, 포맷된 문서를 정의하였고 처리되도록 하였다. 또한 대화형 다이얼로그 기능을 통해 템플릿을 편집하고 포맷팅 결과를 즉시 화면에서 확인할 수 있는 기능을 제공하도록 시스템을 구현하였다.

본 시스템이 갖는 장점은 첫째, 표준 마크업언어를 사용함으로써 다른 가변문서 출판 시스템과의 정보교환과 웹 기반의 출판 시스템으로 확장이 용이하다는 점이다. 둘째, 문서의 데이터, 구조, 표현을 분리하고 XML 기반 언어로 설계함으로써 구조의 변경 및 확장이 수월하다는 점이다.

미흡한 점으로는 XML의 표준 스타일시트인 XSL-FO를 처리하지 못하며, XLink(XML Linking Language)와 XPointer(XML Pointer Language)와 같은 다른 기술에 대한 설계가 미흡하다. 또한 현재 본 시스템은 결과문서를 HTML, PDF의 두가지 포맷으로만 변환 출력할 수 있으며, 아직 개발 중이므로 실용적인 사용을 위해서는 부가기능에 대한 추가적인 보완이 필요하다.

5. 결론 및 향후 연구과제

최근 다변화 되는 사회에서 출판의 요구는 점점 복잡하고 다양해지고 있다. 기존의 레이아웃 기반의 고품질 인쇄출판이 적합한 분야도 있지만, 개인이나 기업 등의 개별적인 특성에 따라 가변 데이터를 포함하는 문서 출판의 요구도 커져가고 있는 실정이다.

이에 본 논문에서는 문서편집 전문가 뿐만아니라 일반인도 가변 데이터 문서를 쉽게 편집할 수 있도록 WYSIWYG 방식의 편집 기능을 지원하는 동적 편집과 포맷팅 방식을 설계 및 구현 하였다.

본 시스템의 장점은 W3C에서 제안하는 XML 관련 표준들을 수용함으로써 다른 시스템과의 정보교환을 용이하게 하고, 시스템의 구조 변경 및 확장이 수월하다는 장점이 있다.

본 논문에서 제안된 시스템은 계약서, 보험증권, 명함, 브로셔, 학습보조자료 등과 같은 개별적인 특성에 따라 가변적인 데이터를 갖는 맞춤형 문서제작을 위한 출판에서 유용하게 사용되리라 사료된다. 또한 XML을 기반으로 하는 가변문서 출판에서 대화형 편집방식과 포맷팅 기능이 결합된 시스템 개발을 위한 연구 모델로 사용될 것이라 기대된다.

향후 연구과제로는 XML의 표준 스타일언어인 XSL-FO에 대한 포맷팅 객체의 편집 방식과 포맷터의 개발이 필요하다. 또한 본 시스템은 템플릿의 동적 편집과 동적 포맷터의 개발에 초점이 맞춰어져 있기 때문에 실용적으로 사용되어지기 위해서는 부가 기능에 대한 추가적인 보완이 필요하다. 그리고 현재 웹 기반의 동적 출판에 대한 중요성이 커져가고 있기 때문에, 문서 편집자과 문서 작성자의 역할에 기반한 저작 도구와 처리 방식에 대한 연구가 필요할 것이다.

참고 문헌

[1] Nathan Hurst, Wilmot Li, Kim Marriot, "Review of Automatic Document Formatting," In Proceedings of the 2009 ACM Symposium on Document Engineering, Sep. 2009.

[2] John Lumley, Roger Gimson, Owen Rees, "A Framework for Structure, Layout & Function in Documents," In Proceedings of the 2005 ACM

Symposium on Document Engineering, Nov. 2005.
[3] John Lumley, Roger Gimson, Owen Rees, "Configurable Editing of XML-based Variable-Data Documents," In Proceedings of the 2008 ACM Symposium on Document Engineering, Sep. 2008.
[4] W3C, XSL Transformations(XSLT) Version 2.0, http://www.w3.org/TR/xslt20/, 2007.
[5] W3C, XML Path Language(XPath) 2.0, http://www.w3.org/TR/xpath20/, 2007.
[6] W3C, Extensible Stylesheet Language(XSL) 1.1, http://www.w3.org/TR/xsl11/, 2006.
[7] UCanCode.NET, Variable Data Printing Solution, http://www.ucancode.net. 2010.
[8] Quint V., Vatton, "Technique for Authoring Complex XML Documents," In Proceedings of 2004 ACM Symposium on Document Engineering, Oct. 2004.
[9] Villard, L and Layaïda, N, "An Incremental XSLT Transformation Processor for XML Document Manipulation," In Proceedings of the 11th World Wide Web Conference, Hawaii, 2002.
[10] 윤상민, “큰 XML 문서를 위한 점진적인 XSLT 처리”, 한국과학기술원 석사학위논문, 2003.
[11] 박성진, “XML 문서 변경 탐지 기능을 갖는 통합 리파지토리 시스템”, 한국산학기술학회논문지, Vol. 10. No. 10, pp.2696-2707, 2009.
[12] 공용해, 김명숙, “XML 정보검색의 효율적 전처리를 위한 문서여과 알고리즘”, 한국산학기술학회논문지, Vol. 6, No. 1, pp.1-11, 2005.

임 광 택(Kwang-Taeg Lim) **[정회원]**

- 1989년 3월 : 광운대학교 전자계산기공학과 (공학석사)
- 1993년 8월 : 광운대학교 전자계산기공학과 (공학박사)
- 1997년 9월 ~ 현재 : 호원대학교 컴퓨터게임학부 교수

<관심분야>
문서정보처리, 전자출판시스템, XML 응용