구조화 문서 작성/편집 시스템의 설계 및 구현

장명길, 이혜란, 이재성, 주종철, 박동인
국어공학센터/시스템공학연구소

The Design and Implementation of a Structured Document Author/Editor

Myung-Gil Jang, Heran Lee, Jea-Sung Lee, Zongcheol Zhoo, Dong-In Park
Center for Korean Language Engineering / SERI

요 약

본 논문은 문서 작성시 해당 문서의 구조적 제한 조건을 명시한 DTD(Document Type Definition) 문법 및 SGML 태깅 방법에 익숙치 않은 사용자가 문서 템플리트를 통해 오류없는 SGML문서를 작성할 수 있도록 하는 구조화 문서 작성/편집 시스템을 소개한다. 본 시스템은 크게 DTD 해석기와 문서편집기로 구성되며 현재 오브젝트 지향 방법론에 따라 구현 중이다.

1. 서론

ISO 8879에 의해 제정된 SGML(Standard Generalized Markup Language)은 서로 다른 문서처리 시스템간의 문서교환이 가능하도록 한 표준화된 기술언어로서 이를 통해 문서를 논리적인 구성요소로 구조화시킬 수 있을 뿐 아니라 각 문서 구성요소의 특성 및 요소간의 링크관계 등을 정의할 수 있다.

그러나 이러한 구조화된 문서 즉, SGML문서는 해당 문서의 구조를 정의하는 문서 타입 정의부 즉, DTD(Document Type Definition)에 따라 작성규칙이 바뀌기 때문에 일반 워드프로세서로는 정확한 SGML문서를 작성하기 어렵다. 따라서 문서 작성자가 일반 워드프로세서로 문서를 작성하듯이 단순히 문서 내용만을 입력함으로써 손쉽게 SGML문서를 작성하는 도구가 필요하다.

SGML문서를 생성하는 방법은 대체로 두가지 방법 즉, 기존의 워드프로세서로 작성된 전자문서를 SGML문서로 전환하는 방법과 처음부터 새로운 문서를 SGML문서 전용 작성기로 작성하는 방법이 있다. 본 논문에서는 후자의 방법으로 구조화문서를 작성하고자 할 때 필요한 SGML문서 작성용 편집기에 대해 논하고자 한다.

2장에서는 구조화 문서 작성기가 갖추어야 할 기능을 중심으로 편집기의 요구사항을 살펴보고, 3장에서는 본 시스템의 구성모듈별로 구현방법을 설명한다.

2. 요구사항 정의

일반적으로 SGML문서 작성기(SGML Authoring Tool)는 사용자가 DTD의 작성규칙에 따라 문서를 작성해야 하는 부담을 덜고 문서의 내용에만 관심을 갖고 작성할 수 있도록 편리한 저작환경을 제공하며, SGML 문서의 검증 및 문서교환을 위한 Import/Export기능 등을 지원해야 한다[7].

SGML문서 편집기가 갖추어야 할 요구사항들을 나열하면 다음과 같다[2].

. 편집기에 의해 작성된 문서가 해당 DTD의 규칙에 맞게 생성되어야 한다.
. 문서의 논리적인 구조를 토대로 실제적인 마크업이 이루어져야 한다.
. DTD에 맞게 작성되도록 사용자를 돕고 유도해 주어야 한다.
. 문서 작성자가 현재 작업중인 컨텍스트 상에서 어떤 엘리먼트에 대해 편집이 가능한지를 알 수 있도록 지원하여야 한다. 예를 들면 선택가능한 엘리먼트들을 나열해 줄 수 있어야 한다[3].
. 문서의 논리적인 구조를 사용자에게 보여 줄 수 있어야 한다.
. 논리적인 구조와 그 내용에 대한 조작이 수행될 수 있어야 한다.

DTD의 규칙에 맞게 태그와 애트리뷰트, 문서 내용을 작성함에 있어서 어떤 방식으로 사용자에게 작성 기능을 제공할 것인지는 편집기의 사용자 인터페이스 설계상의 문제로서 본 편집기는 다음과 같은 인터페이스 구조를 따른다.

. 일반 텍스트 편집 기능에 Incremental Parser를 부가한 방식의 편집기 구조를 택하지 않고 문법 구동형(Syntax-driven) 편집기로서 문법에 맞는 문서 구조의 기본 골격을 갖는 템플리트 제공 방식을 택하였다.

. 사용자가 태그와 애트리뷰트를 직접 입력하지 않고 템플리트 및 contextual 메뉴를 통해 문서의 내용만을 입력하는 방식을 채택하였다. 즉, 문법에 관련된 구조는 편집기에 의해 자동으로 생성되거나 메뉴상에서 가능한 편집명령을 선택함으로써 형성되도록 하였다.

3. 구조화 문서 작성/편집 시스템

3.1 기본 구성

구조화 문서 작성/편집 시스템(이하 KLE-OX 시스템으로 명명)은 크게 DTD 해석기 모듈과 구조화 문서 작성/편집기로 구성되며, 그 기본 구성도는 그림 1과 같다.

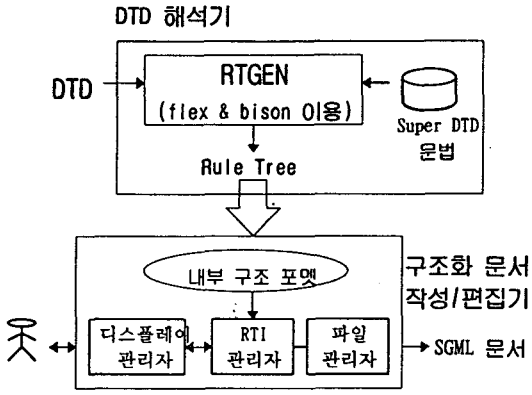

그림 1. KLE-OX 시스템의 기본 구성도

DTD 해석기 즉, RTGEN(Rule Tree GENerator)는 임의의 DTD 파일로부터 그 문서의 구조 정보를 가지는 RT 구조를 생성하는 모듈이다. 이 때 생성된 RT 화일은 SGML 문서를 작성하는 모듈인 구조화 문서 작성/편집기에 입력된다. 이 모듈은 flex와 bison 툴을 사용하여 구현되었다.

구조화 문서 작성/편집기는 RT 구조로부터 만들어지는 문서 템플리트를 이용하여 SGML 문서를 작성/편집하는 모듈이다.

3.2 DTD 해석기

DTD 해석기는 DTD를 해석하기 위하여 Super DTD 문법을 정의한다. Super DTD 문법은 ISO 8879에서 정의된 DTD를 기술하기 위한 메타 문법으로 이 문법에 따라서 다양한 종류의 DTD가 작성된다. 본 연구에서는 TEI P2[9]에서 정의한 DTD 문법을 Super DTD 문법의 기초로 사용하였다.

DTD 해석기에서는 SGML 선언부와 DTD 문법의 규칙들을 lex spec.과 yacc spec.에 작성한다. 특히, yacc spec.에서 각 rule의 semantic action 부분에는 DTD를 파싱한 정보 즉, DTD 검증의 결과를 보여주는 루틴과 DTD 규칙 정보로부터 RT 구조를 생성하는 루틴을 기술한다.

다음은 입력 DTD의 엘리먼트 선언부에서 문서 구조를 표현하는 content model 부분을 인식하는 yacc rule과 RT를 구성해 가는 코드들이다.

```
model    : '(' tokengrp ')'
                { $$ = $2; }
         | '(' tokengrp GRPOPT
                { $$ = MakeOccNode($2, '?'); }
         | '(' tokengrp GRPPLUS
                { $$ = MakeOccNode($2, '+'); }
         | '(' tokengrp GRPREP
                { $$ = MakeOccNode($2, '*'); }
         ;
tokengrp : seqtokens { $$ = $1; }
         | ortokens { $$ = $1; }
         | andtokens { $$ = $1; }
         ;
seqtokens : token  { $$ = $1; }
         | token ',' token
                { $$ = MakeCnnNode($1, $3, ','); }
         | seqtokens ',' token
                { $$ = AddSibNode($1, $3); }
         ;
ortokens : token '|' token
                { $$ = MakeCnnNode($1, $3, '|'); }
         | ortokens '|' token
                { $$ = AddSibNode($1, $3); }
         ;
andtokens : token '&' token
                { $$ = MakeCnnNode($1, $3, '&'); }
         | andtokens '&' token
                { $$ = AddSibNode($1, $3); }
         ;
token    : PCDATA  { $$ = MakeTrmNode(-1); }
         | NAME
           { temp = MakeTrmNode(GetNameTblIndex()); }
           occurrence
                { if($3 == -1) $$ = temp;
                  else $$ = MakeOccNode($3, temp); }
         | model   { $$ = $1; }
         ;
occurrence :       { $$ = -1; }
         | '?'     { $$ = '?'; }
         | '*'     { $$ = '*'; }
         | '+'     { $$ = '+'; }
         ;
```

위 규칙으로부터 생성되는 RT의 자료구조 형태는 일반 트리(general tree) 구조이다. 일반 트리 구조는 DTD의 문서 구조를 적절히 이 구조에 반영할 수 있고 그 문서의 인스턴스도 쉽게 표현할 수 있다. 또한, 이후에 그 문서의 구조를 이용한 SGML 문서의 작성/편집에서 일반 트리 구조를 탐색함으로써 적절한 정보의 획득/갱신과 아울러 트리 구조의 변경을 쉽게 행할 수 있다.

'메모' DTD로부터 생성하는 RT 구조의 예는 그림 2와 같다. 이 그림에서 보는 바와 같이 RT를 구성하는 노드는 Tag 노드, Control 노드, Data 노드, Declared 노드, Root 노드 등 5 가지 종류가 있다. 특히 Control 노드는 구조화 문서 작성/편집기에서 문서 구성요소를 작성하는 순서를 제어하는 정보를 가진 중요한 역할을 하는 노드이다.

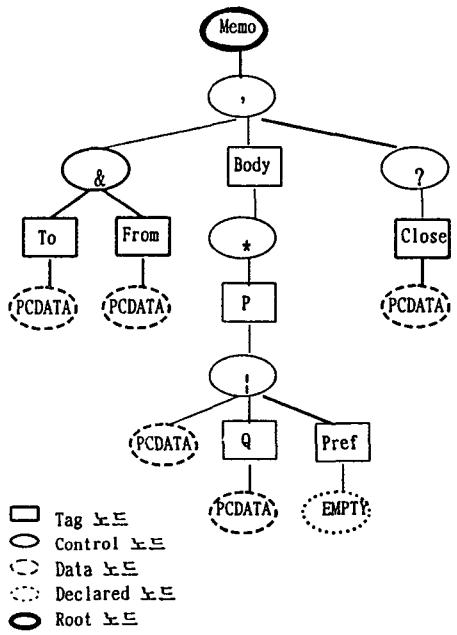

그림 2. '메모' DTD의 RT 구조

3.3 구조화 문서 작성/편집기

3.3.1 개요

본 편집기는 오브젝트 지향 방법론에 따라 설계되었으며 크게 RTI(Rule Tree Instance) 관리자, 디스플레이 관리자(Display Manager), 파일 관리자(File Manager)의 3가지 모듈로 구성되어 있다. 따라서 각각의 모듈은 자체의 자료구조 및 함수들을 가지되, 다른 모듈간의 상호 의존관계는 메세지를 주고 받음으로써 이루어진다.

모듈간의 상호작용은 크게 초기화, 커서이동, 태그 및 데이타 편집, 파일입출력, 구조편집 명령처리

등, 5가지 이벤트에서 주로 발생한다. 그림 3은 초기화 이벤트에서 발생하는 모듈간의 상호관계를 메세지 흐름으로 예시한 것이다.

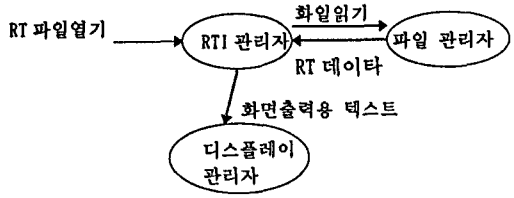

그림 3. 초기화를 위한 모듈간의 메세지 흐름

3.3.2 RTI 관리자

RTI 관리자는 KLE-OX 시스템의 핵심 모듈로 RTI를 관리한다. RTI 관리자의 대표적인 기능으로는 RTI 초기화, RTI 탐색, RTI 수정, RTI 상태변경, RTI 명령처리, RTI 저장 등이 있다.

가. 'RTI 초기화' 모듈

RTI 초기화 모듈은 RTI 관리자가 RTI를 KLE-OX 시스템에 처음 로드하는 루틴이다. 이것은 RTGEN으로부터 생성된 RT 파일을 KLE-OX 시스템으로 처음 로드하거나 외부의 SGML 문서를 편집기 내부로 임포트할 때에 동작한다.

나. 'RTI 탐색' 모듈

RTI 탐색 모듈은 RTI를 탐색하는 루틴으로서 RTI 관리자가 여러 기능들을 수행하는 데 필수적인 기본 기능이다. 예를 들어, 이 루틴은 RTI를 템플리트 텍스트(디스플레이 관리자가 화면에 출력할 수 있는 형태)로 만들 때 적절히 이용된다. 탐색방법은 RTI 구조의 root 노드에서 시작하여 pre_order & depth_first 탐색 방법으로 RTI 탐색의 다양한 목적에 맞는 서브트리을 찾아가는 것이다. 이때, RT 노드 중에 Control 노드에 대한 해석이 중요한 역할을 한다.

다. 'RTI 수정' 모듈

RTI 수정 모듈은 디스플레이 관리자로부터 RTI 수정에 관한 정보를 넘겨 받아 그 기능을 수행하는 루틴이다. 예를 들어, 트리 위치 정보, 변경 명령 정보, 변경할 데이타 등을 가지고 RTI 구조 및 문서 데이타에 대한 수정 작업을 수행한다.

라. 'RTI 상태변경' 모듈

RTI 상태변경 모듈은 디스플레이 관리자에 의해 제어되는 커서 위치 정보(Current Cursor Position: CCP)가 RTI 관리자에 의해 제어되는 RTI상에서의 위치 정보(Current Rule Tree Position: CRTP)에 반영되도록 함으로써 두 모듈간의 현재 상태를 일치시키는 역할을 한다.

마. 'RTI 명령처리' 모듈

RTI 명령처리 모듈은 화면상에서 임의의 커서 위치에서 사용자에 의해 선택이 가능한 명령어들에 대한 정보를 알아내기 위한 루틴으로서 RTI 관리자는 디스플레이 관리자의 요구가 있을 때마다 가능한 명령어 리스트를 디스플레이 관리자에게 전달하여야 한다. 예를 들어, CCP 정보에 따라서 원하는 태그를 입력 또는 삭제해야 하는 경우에 수행된다.

바. 'RTI 저장' 모듈

RTI 저장 모듈은 RTI 관리자가 파일 관리자와 상호작용하면서 RTI 파일들을 저장하는 문제를 다룬다. 이 모듈에서 저장하는 파일 포멧으로는 2 가지가 있는데, KLE-OX 시스템의 내부 파일인 OXF 포멧과 문서 익스포트를 위한 아스키 파일 포멧이다. 첫째, OXF 포멧은 RTI 내용을 바이너리 형태로 저장한 것이다. 이것은 RTGEN에서 생성한 RT 파일 포멧과 거의 같고 KLE-OX 편집기에서 작성한 문서 인스턴스의 내용이 추가되어 있을 뿐이다. 둘째, 아스키 포멧은 일반적인 SGML 문서 파일 형태인 fully tagged 문서로 만들어 진다. 이것은 SGML 문서가 특정 디바이스에 의존되지 않고 문서 교환에 이용될 수 있도록 하는 표준 포멧이다.

3.3.3 디스플레이 관리자

디스플레이 관리자는 RTI 관리자로부터 화면에 출력할 내용을 받아 편집기의 화면에 보여주고, 사용자

로부터 편집에 관련된 각종 조작 명령을 받아 이를 다시 RTI 관리자에게 알려주는 역할을 한다. 따라서 이 모듈은 사용자 인터페이스에 관련된 거의 모든 기능을 비롯하여 편집기가 갖추어야 할 기본적인 편집 기능들을 모두 포함하고 있어야 한다.

문서 작성자가 새로운 구조화문서를 작성하고자 할 때 태그와 애트리뷰트를 직접 입력하지 않고 템플리트 및 contextual 메뉴를 통해 문서 내용만을 입력하는 방식으로 구현하기 위해 디스플레이 관리자가 갖추어야 할 기능은 다음과 같다.

. RTI 관리자로부터 올바른 문서구조를 갖는 템플리트 텍스트를 받아서 이를 화면에 출력한다. 이 때 사용자의 입력부분에 해당하는 placeholder는 가상 텍스트로 보여 주고 있다가, 작성자의 입력문자가 들어오는 즉시 이 실제 데이타로 교체되도록 한다. 또한 사용자에 의한 문자의 삽입, 삭제는 화면상에서 태그가 아닌 실제 데이타 영역에서만 이루어져야 한다.

이러한 편집 가능영역에 대한 구별은 템플리트 뿐만 아니라 이미 작성된 모든 SGML문서를 편집할 때도 동일하게 적용된다. 따라서 태그와 문서 데이타에 대한 차별화된 편집을 제공하기 위해 다음과 같은 점들을 고려하여 구현하였다.

- 커서 이동 : 태그와 문서 데이타 이외의 영역으로는 커서가 이동하지 않아야 하고, 커서의 위치에 따라 편집 모드가 다르게 제공되어야 한다. 즉, 태그 상에서는 직접적인 편집이 아니라 메뉴를 통해 가능한 태그 리스트 중에서 사용자가 선택하도록 한다.

- 편집 기능 : 편집 모드의 결정 및 가능한 태그 리스트 등의 출력은 RTI관리자와의 상호작용을 통해 이루어지며, 실제 화면상의 출력 양식을 제어하기위해 디스플레이 관리자만의 독자적인 자료구조로서 텍스트 버퍼를 갖는다.

- 화면 출력 : 태그와 가상 문서 데이타, 그리고 사용자가 입력한 데이타를 구분함으로써 다른 양식으로 화면에 출력할 수 있도록 하고, 커서의 위치 및 스크롤에 따른 변화는 내부 텍스트 버퍼의 인덱스를 통해 출력양식이 제어되도록 하였다.

그림 4는 사용자가 [파일]메뉴에서 [템플리트 파일 열기]명령을 선택하여 '메모' 문서를 처음 작성할 때 수행한 예로서, 디스플레이 관리자가 템플리트 텍스트를 화면에 출력한 결과이다.

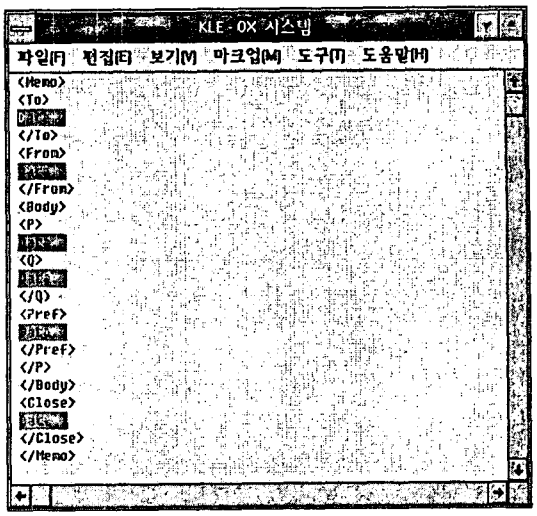

그림 4. 템플리트 텍스트의 초기 화면

3.3.4 파일 관리자

파일 관리자는 SGML 문서를 작성하는 사용자 인터페이스의 설계에 따라 정의된 [파일] 메뉴와 그 외의 메뉴에서 발생하는 파일의 관리를 총괄하는 모듈이다. 그림 5 는 파일 관리자의 동작환경을 나타낸다.

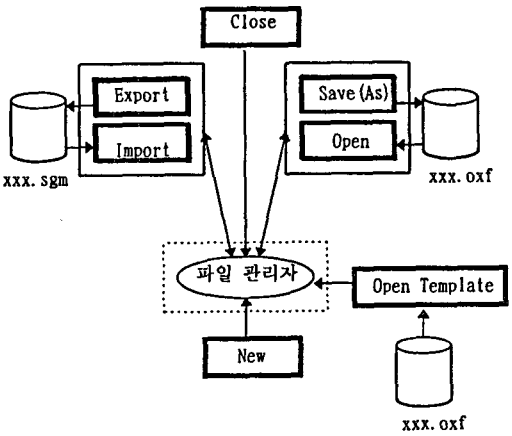

그림 5. 파일 관리자의 동작 환경

'□' 부분은 파일 관리자가 사용자와 인터페이스하면서 관리하는 세부메뉴들을 나타낸다.

4. 결론

SGML문서 작성 시스템은 어떠한 문서 구조를 갖는 문서형태(DTD)에 대해서든지 사용자가 그 구조를 갖는 문서를 그 내용과 상관없이 다양하게 만들 수 있도록 하며, 이미 작성된 구조화문서에 대해서는 사용자가 그 구조를 잘 알고 부가적인 편집을 가능케 하며, 이를 다시 다른 시스템에서도 사용할 수 있도록 교환 포멧으로 변환시켜 주기도 한다.

이러한 시스템을 위해 DTD해석기와 문법구동형 편집기의 기본 모듈을 구현하였으며, 편집기의 각 모듈은 현재 개발이 진행 중이다. SGML문서는 교환 및 문서 구조화의 목적이 크기 때문에 한글이나 영어에만 한정되지 않고 다국어 문서에 대한 지원 방법, 새로운 저작 행위 및 구조 정보를 잘 반영한 사용자 인터페이스, 표준화된 포멧팅 정보를 위한 DSSSL지원 기능 등에 대해 계속 연구되어야 한다. 특히 최근 이용이 확대되고 있는 HTML문서의 저작 환경과 관련된 시스템 지원기능도 요구된다.

참고 문헌

[1] ISO/IEC TR 10037, *Infomation technology - SGML and Text-entry Systems - Guidelines for SGML Syntax-Directed Editing Systems*, 1991

[2] Ch. Boitet (ed), *PreCOLING-92 TUTORIALS*, GETA(IMAG), Nantes, 1992

[3] Joan M. Smith, *SGML AND RELATED STANDARDS*, document description and processing languages, Ellis Horwood Limited, 1992

[4] Martin Bryan, *SGML An author's guide to the Standard Generarized Markup Language*, Addison-Wesley, 1988

[5] R. Furuta, V. Quint, and J. Andre, "Interactively editing stuructured documents", *Electronic Publishing, Vol. 1(1)*, 19-44, April, 1988

[7] SoftQuad Inc, *SoftQuad Author/Editor Version 3.0 for Motif*, March, 1994.

[8] Goldfarb, C.F., *The SGML Handbook*, Oxford Univ. Press, 1990.

[9] C.M. Sperberg-McQueen, Lou Burnard, Guidelines for Electronic Text Encoding and Interchange, *TEI P2, Chapter 42, Draft Version 2*, TEI, 1992

[10] Thomas Reps, Tim Teitelbaum, and Alan Demers, "Incremental Context-Dependent Analysis for Language-Based Editors", *ACM Trans. on Programming Languages and Systems, Vol.5, No3*, July 1983

[11] Tim Teitelbaum and Thomas Reps, "The cornell program synthesizer: A Syntax-directed programming environment", *CACM, 24(9)*:563-573, September 1981

[12] T. F. Lunney and R. H. Perrot, "Syntax-directed editing", *Software Engineering Journal*, March 1988