정보학연구 제2권 제2호 1999년 11월(pp.35-49)
The Studies of Information Technology, Vol.2, No.2

SGML 문서형 정의부 작성을 위한 대화형 편집 시스템 설계 및 구현**

김창수/정희경*

요 약

본 논문에서는 ISO 8879(Standard Generalized Markup Language)에 근거하여 문서의 논리 구조를 정의하고 있는 DTD(문서형 정의부 : Document Type Definition)를 자동 편집하기 위한 시스템을 설계 및 구현하였다.
SGML DTD는 문서의 논리 구조 유형을 정의하는데 그 구조가 복잡하여 일반 텍스트 편집기로 작성하기가 쉽지 않다. 그러므로 일반 사용자도 쉽게 사용할 수 있도록 그래픽적으로 윈도우 환경하에서 트리 형태로 표현하여 직접 조작함으로써 복잡한 DTD를 쉽게 작성할 수 있는 DTD 자동 생성 및 편집하는 시스템에 대해 연구하였다.
이를 위해 SGML을 중심으로한 문서 구조를 분석하고, SGML 문서처리 모델을 연구하였으며, SGML DTD를 구성하는 엘리먼트(element), 엔티티(entity), 속성(attribute), 표기법(notation), 주석(comment) 등의 논리 구조를 윈도우상에 트리 형태로 표현하였다.

Ⅰ. 서론

정보화 사회 및 인터넷(internet)이 보편화되고, 개방 환경하에서 멀티미디어(multimedia) 문서를 효율적으로 상호 교환, 저장 및 검색의 중요성이 날로 증대 되어감에 따라 주변장치, 응용, 네트워크 등과 독립적으로 처리될 수 있는 환경이 요구되고 있다[6,7,13].

이러한 요구를 위해 국제 표준 협회에서 제정된 규격으로 ODA(Open Document Architecture)와 SGML이 있다[8,13,16]. ODA는 문서 구조가 제한되어 복잡한 문서의 표현보다는 간단한 문서 작성에 적합하고, SGML은 복합 문서에 대해 마크업의 개념만을 정의한 메타 언어(meta language)로 복잡한 문서를 작성하는 것뿐만 아니라 서로 다른 시스템간에 효율적으로 또한 정보를 전달할 수 있는 장점을 갖는다. 초기의 마크업은 문서에 본문 이외의 식자 코드를 추가하여 시스템이 이를 인식하여 처리하는 포맷팅의 역할로 사용되었으나, 점차로 문서의 구조와 속성을 표현하는 방향으로 변화하였다. 즉 문서의 논리 구조와 속성을 기술함으로써 다양한 형태로 표현이 가능하게 되었다.

SGML은 이미 미 국방성의 CALS(Commerce At the Light Speed)와 디지틀 도서관 등의 다양한 응용 분야에서 정보 처리를 위한 문서 기술 표준으로 채택되었으며, 미국 및 유럽 공동체 등에서 출판 관련 분야에서 사용되고 있으며, 또한 미국 화학 학회, 수학 학회 등에서 사용하고 있다. 일본에서는 이미 통산성 주도하의 일본 표준을 제정 공표 하였으며 1990년부터 특허

* 배재대학교 컴퓨터공학과 박사과정/조교수
** 이 논문은 1998년 과학재단의 특정기초 연구과제 연구비에 의하여 연구되었음.

공문의 전자화에 쓰여졌고, SGML을 기반으로 한 학술 논문의 데이타베이스화의 검토가 시작되는 등 세계적으로 SGML의 보급에 힘쓰고 있다. 이 처럼 SGML이 복합 문서 처리의 중요한 기반 기술로 부각되고 있는 실정이다.

SGML을 기반으로 하는 응용 기술이 이미 선진 각국에서는 SGML 문서를 편집하고 관리할 수 있는 많은 상용 도구들이 개발되어 왔다. 특히 문서형 정의부(DTD)는 SGML 문서의 논리적 구조가 내포된 부분으로서 SGML 문서 작성에서 매우 중요한 부분으로 전문가에 의해 작성되는데 일반인도 쉽게 이해하고 편집할 수 있도록 응용 프로그램의 개발이 요구된다. 이에 따라 외국의 경우 다양한 도구들이 소개되고 있지만 문화적 영향에 따른 차이점 특히 한글 처리 등의 문제점이 대두되고, 국내에서는 SGML 문서 저작도구, 문서형 정의 저작도구 등의 프로토 타입 수준의 개발이 있었지만 아직 미비한 실정이다[1,12].

따라서 본 논문에서는 ISO 8879에서 문서의 논리 구조를 표현하는 형식에 따라 기술하는 문서형 정의부(DTD)를 편집할 수 있는 SGML DTD 편집 시스템을 설계 및 구현하였다. 문서형 정의부는 그 구조가 복잡하여 일반 텍스트 편집기로는 작성하기가 쉽지 않기 때문에 그래픽 환경하에서 트리 구조 형태로 표현하여 보다 효과적으로 문서형 정의부를 자동 생성 및 편집할 수 있도록 함으로써 일반 사용자가 한글 처리 문제에 구애받지 않고 용이하게 작성할 수 있도록 하였다.

그리고 문서형 정의부를 구성하는 요소인 엘리먼트, 엔티티, 속성, 표기법, 주석 등을 그래픽적으로 표현하고 각각을 편집할 수 있도록 편집 대화 상자를 두었으며, 문서의 구조를 트리 구조 형태로 표현하여 대화적으로 편집할 수 있도록 하였다. 또한 편집한 내용을 수정 보완할 수 있도록 다양한 편집 기능(추가, 삭제, 수정, 숨김 및 복구)을 두어 작성한 뒤, SGML 형식으로 문서형 정의부 파일이 생성되도록 하였고, 생성된 파일을 미세 조정할 수 있도록 텍스트 편집 기능을 두었다. 본 편집 시스템의 구현에서 사용자 인터페이스를 위해 X-Windows와 Motif를 사용하였다[3,5].

II. SGML 문서와 DTD

본 장에서는 SGML 문서의 구성과 문서의 논리 구조를 표현하는 문서형 정의부(DTD)에 대한 기본 개념에 대해 설명하였다.

2.1. SGML 문서의 구성

SGML 문서는 문서에 사용될 언어와 사용될 문자 집합을 선언하고 SGML 문서 처리에 필요한 요소 등을 정하는 SGML 선언부, 문서의 자체를 구성할 수 있는 논리적인 구조와 계층적 구조를 내포하고 있는 문서형 정의부(DTD), 그리고 문서형 정의부에 의해 실제 내용을 포함하는 SGML 실례 문서부의 세 부분으로 구성된다.

2.1.1. SGML 선언부

SGML 문서를 구성하는데 필요한 문자 집합과 코딩 규칙 등을 정의하며, 이 기종 시스템간 전송시 문서가 어떠한 기준으로 작성된 것인지에 대한 정보를 포함한다. 선언부는 문서에 사용할 문자 집합을 정의하는 부분, 처리 능력에

대한 범위를 정의하는 부분, 주요 구체 구문(concrete syntax)을 정의하는 부분, 문서가 갖추어야 할 기능들을 표현하는 부분 등으로 구성된다.

2.1.2. 문서형 정의부(DTD)

문서 자체의 논리 구조를 선언하는 부분으로서 임의 특정 형태를 갖는 모든 문서에 적용하는 마크업 선언들 집합으로 구성되며, 시작 부분에 문서형 정의에 대한 고유 이름을 갖는다. 문서형 정의에 선언할 수 있는 선언들은 문서형 선언(document type declaration), 엘리먼트 선언(element declaration) 및 엔티티 선언(entity declaration)과 속성 선언(attribute declaration)을 들 수 있으며, 이외에 표기법 선언(notation declaration) 및 주석 선언(comment declaration), 단축 참조 선언(short reference declaration) 등이 있다.

(1) 문서형 선언

문서형 정의는 문서의 논리 구조와 기타 정보를 기술하기 때문에 이를 참조할 수 있도록 SGML의 첫 부분에 특수한 마크업 선언을 하는데 이를 문서형 선언이라 하며, MDO(Markup Declaration Open) 기호(<!)로 시작하여 MDC (Markup Declaration Close) 기호(>)로 끝난다. MDO 다음에 DOCTYPE 이라는 예약어와 고유 이름인 공통 식별자를 차례로 기술한다. DOCTYPE 선언의 전형적인 예는 다음과 같다.

```
<! DOCTYPE book [
                .....
]>
```

(2) 엘리먼트 선언

식별자와 내용부로 이루어지며 각각 엔티티를 포함할 수 있고, 또한 부엘리먼트를 포함할 수 있다. 먼저 공통 식별자에 이어 차례로 엘리먼트 선언을 해야 하며, 임의의 엘리먼트 선언은 내용 부분에 다른 엘리먼트의 공통 식별자를 가질 수 있는데 이 공통 식별자를 갖는 엘리먼트는 선언중인 엘리먼트의 부엘리먼트가 된다. 부엘리먼트에 대한 선언도 필요하며, 트리 형태의 구조를 갖는 문서형 정의부에 있는 엘리먼트 선언들에 의해 문서 구조를 파악하게 된다. 엘리먼트 선언 방법은 다음과 같다.

```
<!ELEMENT memo - O ((TO &
    FROM),BODY,CLOSE?) + (FN) >
```

위의 예에서 ELEMENT는 엘리먼트 선언의 예약어이고, memo는 엘리먼트명이다. 다음으로 엘리먼트에 대한 시작 태그 및 종료 태그를 생략할 지에 대해 나타내기 위해 두 문자를 사용한다. '-'는 생략이 가능하지 않음을 나타내고, 'O'는 생략 가능함을 나타낸다. 다음으로 엘리먼트에 대한 내용을 나타내는데, 이때 발생 지시자와 연결자를 사용하게 된다. 발생 지시자는 opt(? : 부엘리먼트가 0번이나 1번 사용됨), plus(+ : 1번 이상), rep(* : 0번 이상)가 있으며, 연결자는 seq(, : 앞뒤의 부엘리먼트가 반드시 존재하되 이 모델 그룹에 대한 순서에 따라 발생), or(| : 연결자 전후 부엘리먼트가 둘 중 하나나 두 부엘리먼트가 발생), and(& : 연결자 전후의 부엘리먼트가 순서에 상관없이 둘다 발생)가 있다. 또한 포함(inclusion)과 제외(exclusion)의 두 가지 예외를 선언하는데 포함은 - 기호를 제외는 + 기호를 사용한다. 위의 예에서는

memo라는 엘리먼트에 FN이라는 엘리먼트의 포함을 나타낸다.

(3) 엔티티 선언

문서내에서 참조될 수 있는 문자 집합의 단위로서 문서를 반복해 입력할 필요가 있을 때 참조할 수 있는 엔티티들을 정의하여 사용한다. 이는 일반(general) 엔티티와 파라메타(parameter) 엔티티로 나누어지며, 일반 엔티티는 문서 구조상의 단순한 문자들을 가지고, 파라메타 엔티티는 교체될 문자가 다른 엘리먼트의 공통 식별자 즉 엘리먼트 이름이 되는 경우로서 문서형 정의부 내에서만 사용된다. 그리고 일반 엔티티에서 교체될 엔티티 문자가 현재 처리중인 문서 내에 존재하지 않는 엔티티를 외부(external) 엔티티라 한다. 엔티티 선언 방법은 다음과 같다.

```
<!ENTITY CALS "Commerce At the Light
    Speed">
```

이렇게 선언된 내용을 실제 문서에서 사용 시에는 &CALS;에 의해 참조된다. 또한 매개변수 엔티티의 예는 다음과 같다.

```
<!ENTITY %html.content
      (HEAD, BODY+) >     -- 선언 예
<!ELEMENT HTML O O
      (%html.content;) >  -- 참조 예 --
```

(4) 속성 선언

문서 엘리먼트에 관련되어 있는 속성들과 그 속성들이 가질 수 있는 값들을 정의하는 것으로, 문서 혹은 엘리먼트의 상태, 문서 텍스트가 출력되는 형식, 외부로부터 데이터가 문서에 추가될 경우에 그 위치 및 크기 등을 알려준다. 속성 지정을 하기 위해 속성은 이름과 속성 값을 갖는다. 속성 선언은 다음과 같으며, 여기서 memo는 관련 엘리먼트 명이며, status는 속성명이고, 다음에는 가능한 속성 값이 오며, 마지막에 디폴트(default) 값을 나타낸다.

```
<!ATTLIST memo status
      (confiden | public) public >
```

(5) 표기법 선언

속성 값에 선언된 NOTATION은 SGML 문서에 포함되어 있는 데이터 중 수학 공식 및 그래픽, 포맷된 텍스트, 2진 데이터, 비트맵, 멀티미디어 데이터 등 특수한 부호와 표기법을 사용하는 데이터들을 표현하는데 사용되며, 입력된 내용을 갖는 엘리먼트를 식별하는데 사용된다. 이러한 방법으로 선언된 속성은 엘리먼트 내용을 문서형 선언부 집합에 포함하는 NOTATION들 중 하나와 연관시킬 수 있다.

(6) 주석 선언

주석 선언은 실제 문서의 구조나 내용과는 무관하며 단지 설명을 필요로 하는 경우 사용되며 사용 예는 아래와 같다.

```
<!-- SGML 문서형 정의부 작성을 위한
      ..... -->
```

2.1.3. SGML 문서부

문서의 내용을 정의하는 것으로서 문서간의

상호 연결된 요소들로 구성되며, 각 문서의 요소는 특정 목적을 갖는 문자들을 포함하고, 그 중 특수한 문자나 단어는 여러 다른 문서 요소의 구성 성분이 될 수도 있다. SGML 문서에 대한 마크업 기술 방법과 문서형 정의부에 따라서 SGML 문서를 작성된다.

Ⅲ. SGML DTD 편집 시스템의 설계

본 장에서는 SGML DTD를 설계하기 위해 필요한 기본적인 원칙과 생성 과정 및 설계, X-Windows 환경하에서의 내부적인 트리 데이터의 구현 및 저장 방법 등의 알고리즘에 대하여 설명한다.

3.1. 기본 원칙

SGML DTD 편집 시스템 설계 시에 다음과 같은 기본 원칙을 정하여 일반 사용자로 하여금 효율적으로 SGML DTD를 생성할 수 있도록 하고자 한다.

- 일반 사용자는 문서의 논리 구조를 가지고 있는 DTD 문서만을 보고 전체적인 구조를 파악하기는 어렵다. 따라서 문서를 그래픽 환경하에서 트리 형태로 구현함으로써 쉽게 DTD를 이해하고 추가, 수정, 삭제, 검색, 변경 등의 여러 가지 편집 기능을 이용할 수 있도록 하여야 한다.
- 각 엘리먼트 간에 성립되어 있는 계층 구조를 정확히 파악해야 한다.
- 새로운 데이터를 추가할 경우 메뉴에서 필요한 항목을 선택하여 사용하는 방법보다는 사용자의 편의를 위해 간단히 툴바(toolbar)를 사용하여 추가하고자 하는 곳을 마우스로 선택하여 편집할 수 있어야 한다.
- 엘리먼트의 부엘리먼트를 서로 그룹화하여 모델 그룹 (model group)으로도 선언할 수 있도록 해야 한다.
- 속성 목록, 외부 엔티티, 표기법, 주석, 그룹핑(grouping) 등의 참조는 DTD 트리와는 별도 윈도우를 사용하여 트리의 복잡성을 줄일 수 있어야 한다.
- 엘리먼트의 부엘리먼트를 그룹화 할 수 있어야 한다.
- 발생 빈도 또한 변경이 가능해야 한다.

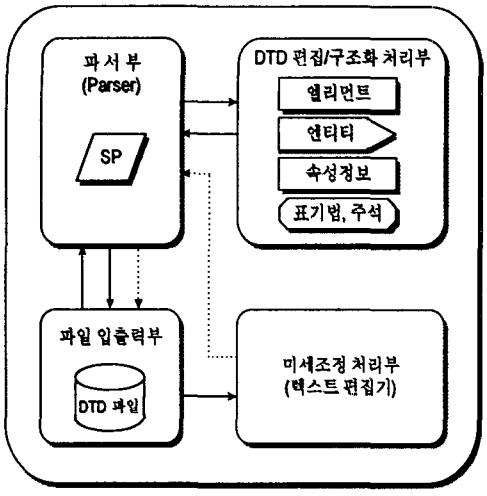
따라서 본 논문에서는 이러한 기본 원칙에 따라 문서형 정의부를 설계 및 구현하였다.

3.2. 시스템 설계

편집 시스템의 전체적인 구성은 그림 1과 같으며 각각 파일 입출력부, 파서(parser)부, DTD 편집 및 구조화 처리부, 미세 조정 처리부 등으로 나누어진다.

3.2.1. 파일 입출력부

새로운 DTD 및 이미 생성되어 있는 파일의 입출력에 관련되며, 파서부와 미세조정 처리부에서 사용된다. 미세 조정 처리부에서 편집된 파일은 파서부에서 분석되며, 이상이 없을 시 파일로 재 저장된다.



(그림 1) 전체적인 시스템 구성

3.2.2. 파서부

본 논문에서는 파서부의 파서는 공용화된 SP 파서를 사용하는데, 이는 ISO 8879 규격을 따르고 한글 처리가 가능하며, DTD가 SGML 문법에 타당한지를 검색하고, 논리 구조의 특성에 맞게 구성되었는지를 검색하여 오류 발생시 이를 화면에 출력해 준다.

3.2.3. DTD 편집 및 구조화 처리부

이는 새로운 파일을 생성할 수도 있고, 이미 생성된 DTD의 경우 SP 파서에 의해 검색된 DTD를 내부 데이터 구조에 의해 구조화된 포인터(pointer)로 재배치한다. 선언된 엘리먼트, 엔티티, 속성 정보, 표기법, 주석 등은 편집 가능한 하나의 데이터로서 존재하며, 이들이 모여 트리를 형성하게 된다. 엘리먼트, 엔티티 이외의 항목은 트리의 복잡성을 줄이기 위해 별도의 윈도우에서 관리되며, 최종 저장 시는 재 조합되어 하나의 DTD 파일로 저장된다.

엘리먼트나 엔티티는 툴바를 이용해 데이터를 추가하고 이름을 수정할 수 있으며, 또한 그루핑, 연결자, 발생 지시자 등을 바로 변경할 수 있으므로 편집 시간을 최대한 활용할 수 있다. 트리가 길어지거나 복잡해질 경우 숨김 기능을 이용해 트리를 간소화할 수 있으며 복구 기능을 이용해 트리를 확장할 수 있다.

3.2.4. 미세 조정 처리부

DTD 파일의 간단한 편집은 자체적으로 텍스트 편집기가 내장되어 있어 필요한 항목을 추가하거나 수정 및 삭제할 수 있다. 변경된 DTD는 파서를 거쳐 이상 유무를 검색후 재 저장된다.

3.3. DTD 트리

DTD에 선언되는 내용은 엘리먼트, 엔티티, 속성 정보, 표기법, 주석 등이 있으며, 이들을 작성하는데 있어 처리되어야 할 구성 요소로는 이름, 모델 그룹, 연결자, 발생 지시자 등이 있다.

(그림 2) DTD 트리

각각의 트리 노드를 연결할 때 상위 노드와의 구분을 위해 연결자가 표시되는 곳으로부터 하위 노드를 연결하도록 하였으며, 새로운 데이터의 추가는 간단히 주메뉴의 툴바를 이용함으로써 사용자의 편의를 도모하였고, 아이콘을 눌렀을 때 커서의 모양이 선택된 아이콘의 모양으로 바뀜을 알 수 있다. 그림 2는 DTD 트리의 간단한 예를 보여준다.

3.3.1. 이름 처리

엘리먼트, 엔티티 및 이름이 필요한 항목은 각각 고유한 이름을 가진다. 이름은 SGML의 제한 사항에 따라 구성하고, 표현되는 형식은 그림 3과 같으며, 엘리먼트 내의 발생 지시자는 툴바의 아이콘을 이용해 바로 변경이 가능하도록 한다.

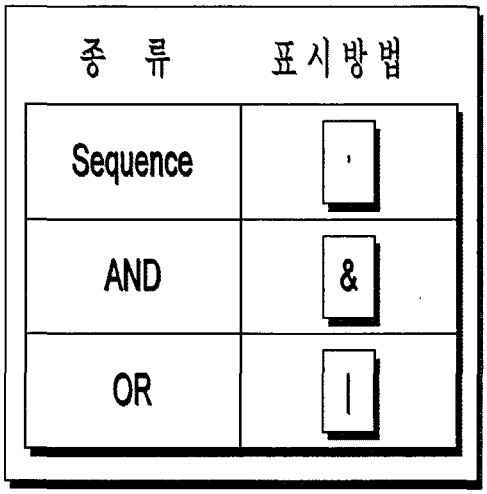
(그림 3) 엘리먼트와 엔티티의 표현

3.3.2. 연결자 처리

연결자는 엘리먼트 간의 상호 관계를 가지며, 부엘리먼트는 하나의 연결자로 연결되는데 연결 지시자의 변경은 단순히 주메뉴의 아이콘을 선택해 변경할 노드를 마우스로 클릭하여 변경한다. 화면에 표시되는 형식은 그림 4와 같다.

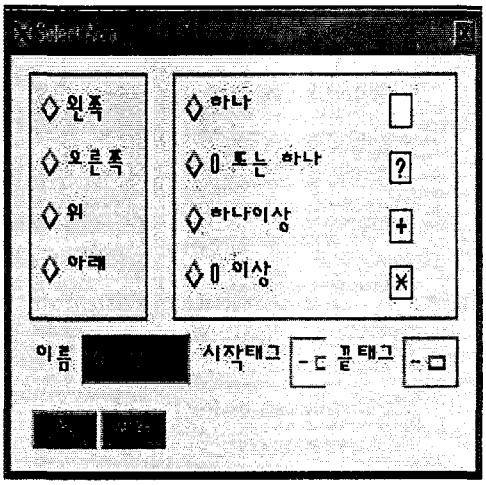
3.3.3. 발생 지시자 처리

모델 그룹에서 각 내용 토큰이나 모델 그룹 전체의 발생에 대한 표시 및 회수를 의미하며, 발생 지시자의 변경도 연결 지시자와 마찬가지로 주메뉴의 아이콘과 마우스를 이용하여 처리



(그림 4) 연결자 처리 표현

된다. 또한 SGML 문서에서 엘리먼트에 대한 시작 태그 및 종료 태그를 생략 가능한지에 대한 정보를 처리하도록 한다. 발생 지시자 처리의 표현은 그림 5와 같다.



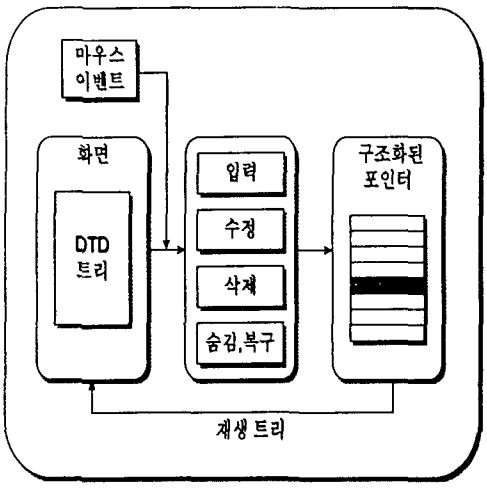
(그림 5) 발생 지시자 처리 표현

3.3.4. 모델 그룹 처리

엘리먼트의 부엘리먼트가 그룹으로 묶여 표현될 수 있는데, 이를 모델 그룹이라 하며 괄호로 묶여져 형성된다. 시스템에서는 발생 지시자와 연결자를 묶어서 처리한다.

3.4. 편집 기능

DTD 트리를 편집하는데 있어 아이콘을 사용해 편집함으로써 편집의 효율성을 높였다. 또한 DTD 트리의 일부를 숨길 수도 있도록 설계하여 복잡한 DTD를 간소화 할 수도 있다. 편집 시스템의 구조는 그림 6과 같다.

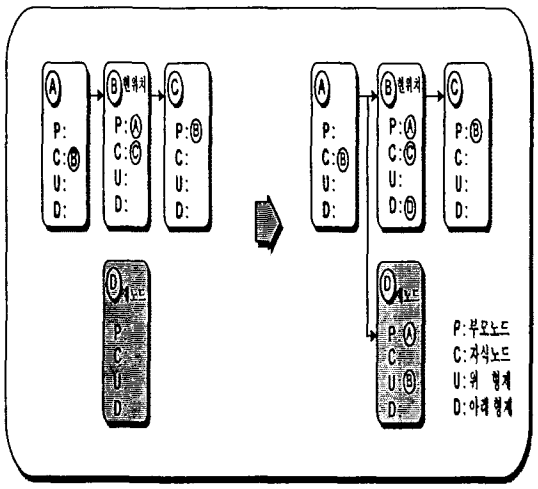


(그림 6) DTD 편집 시스템 구조

3.4.1. 추가 기능

주메뉴와는 별개의 윈도우에서 작업이 진행되며, 추가시 현재 노드에 대하여 노드의 관계(부모, 형제, 자식)를 선택할 수 있게 되어 있다. 추가 기능에는 현재 노드에 대하여 부모 노드 추가는 왼쪽을 선택하고, 자식 노드 추가 시는 오른쪽을 선택하며 형제 노드 추가시 위 형제 노드는 위를 아래 형제 노드는 아래를 선택한다.

아래 형제 노드 추가의 예를 그림 7에 보인다. 아래 형제 노드(아래) 추가시는 현재 노드 B에서 D(아래) 항목에 D가 주어지고, 새로 추가하려는 노드 D의 U(위)에는 현재 위치의 B 노드가 주어진다. 새 노드 D의 세부 항목 중 위형제(U)에 노드 B의 포인터를 가리키도록 하며, B 노드의 세부 항목 중 아래 형제(D)에 노드 D의 포인터를 가리키도록 한다.



(그림 7) 아래 형제 노드(아래) 추가 전과 추가 후

3.4.2. 삭제 기능

더이상 필요로 하지 않는 엘리먼트나 엔티티 및 기타 노드를 삭제하기 위한 기능으로 삭제하고자 하는 항목에 우측 마우스의 클릭(click) 만으로 데이터 표에서 제거되며 삭제를 취소할 경우 이를 복구를 할 수 있다.

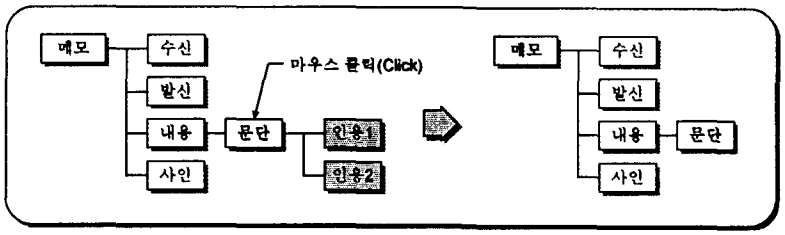
3.4.3. 수정 기능

각 데이터의 이름 및 속성을 수정할 필요가

있을 때, 수정할 데이터에 대해 왼쪽 마우스를 클릭하면 여러 가지 수정 항목 중 하나를 선택하여 데이터를 수정한다.

3.4.4. 숨김 및 복구 기능

복잡한 DTD 트리의 경우 원하는 노드에서 가운데 마우스를 클릭함으로써 현재 노드의 자식 노드들을 숨길 수가 있어 DTD를 간소화 할 수 있으며, 숨겨진 데이터에 다시 가운데 마우스를 클릭 함으로써 한 단계의 자식 노드들을 복구할 수 있다.



(그림 8) 숨김 및 복구

3.5. 내부 데이터 구조

3.5.1. DTD 트리 노드들의 내부적 데이터 구조

DTD 트리를 설계하는데 필요한 내부적인 구조를 정의하는 부분으로 DTD 트리를 구성하고 있는 엘리먼트, 엔티티 및 각각의 속성들에 대한 정보들의 내부적 데이터 구조를 나타낸다.

데이터의 기본 구조는 구조화된 포인터로 되어 있으며, 선언은 그림 9와 같다. 트리에 대한 모든 노드들의 구조를 단순화시키기 위해 그림 9의 동일 구조로 모든 노드를 처리하도록 하였다. 변수는 각각 객체의 이름, 화면에 표시되는 좌표, 태그 처리, 발생 지시자, 데이터 유형, 속성 리스트, 종단 노드의 유뮤, 숨김/복구 등을 저장하고 있으며, 구조내의 구조체 포인터는 형제와 자식 및 위 형제 노드, 아래 형제 노드를 가리킨다.

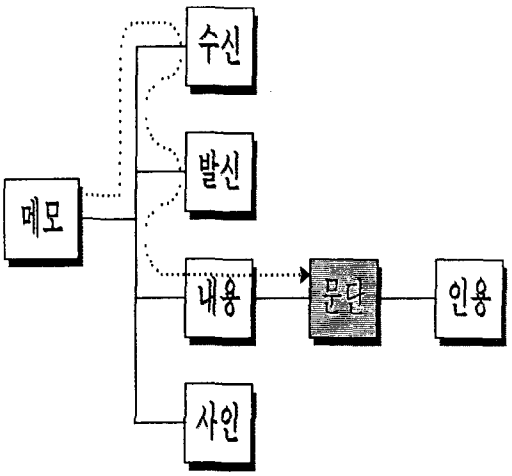
```
typedef struct TreeObject_ST
{
 char *name;                          /* 이름          */
 int   x1, y1;                        /* 좌표1         */
 int   x2, y2;                        /* 좌표2         */
 char *start_tag;                     /* 시작 태그     */
 char *end_tag;                       /* 끝  태그      */
 int   occurrence;                    /* 발생지시자    */
 int   data_type;                     /* 데이타타입    */
 int   attr_list;                     /* 속성유무      */
 int   terminal;                      /* 터미널유무    */
 int   en_disable;                    /* 숨김유무      */
 int   in_exclusion;                  /* inclusion, exclusion */

 struct TreeObject_ST *Parent;        /* 부모          */
 struct TreeObject_ST *Child;         /* 자식          */
 struct TreeObject_ST *Child_Up;      /* 위  형제      */
 struct TreeObject_ST *Child_Down;    /* 아래형제      */
 } TreeObject;
```

(그림 9) DTD 트리의 내부 데이타 구조

3.5.2. 데이터 검색 및 저장

엘리먼트나 엔티티, 그리고 속성 등을 새로이 추가해야 할 경우 추가할 위치를 검색해야 한다. 이를 검색하는 방법에는 깊이 우선 탐색 방식(Depth-first Search)을 이용하였으며, 새로이 생성된 DTD를 저장하는 방법에는 너비 우선 탐색 방식(Breadth-First Search)을 이용하였다.

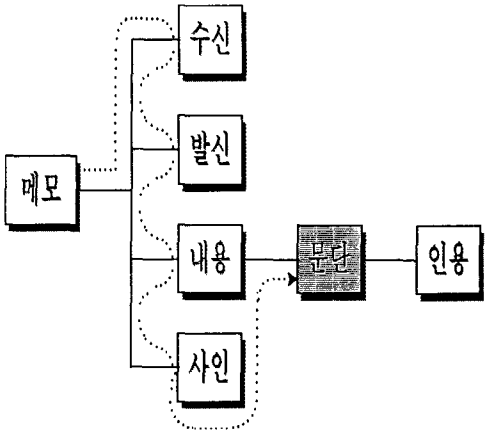
깊이 우선 탐색 방식은 시작 정점으로부터 인접한 정점 가운데 한 정점만을 검색하고 다른 인접한 형제 노드들은 다음에 처리된다. 이어서 자식 노드를 검색해 노드가 있으면 그 중에서 한 정점만을 검색하는 과정이 그 다음에 인접한 정점이 없을 때까지 계속한다. 인접한 정점이 없을 때는 바로 앞에서 방문한 정점의 두 번째 인접한 정점을 방문한다. 이를 위해 하나 이외의 인접한 정점들은 스택(stack)에 보관한다. 예를 들면, 그림10에서 데이터 문단을 검색할 경우 검색되어지는 순서는 수신, 발신, 내용, 문단이다.



(그림 10) 검색 순서

너비 우선 탐색 방법은 지정된 시작 정점부터 검색하고, 이 검색된 정점에 인접한 형제 노드들을 모두 검색한다. 다음에 인접한 정점 중 처음 검색한 정점부터 순서대로 자식 노드들을 찾는다. 이러한 과정을 모든 정점을 방문할 때까지 진행한다. 예를 들면, 그림11에서 데이터 문단을 검색할 경우 검색되어지는 순서는 수신, 발신, 내용, 사인, 문단이다.

화면상에 나타난 데이터들을 저장할 때는 부모 노드로부터 자식 노드로 검색하며 저장해야 하기 때문에 너비 우선 탐색 방법이 적합하다



(그림 11) 저장 순서

3.6. 사용자 인터페이스 (interface)

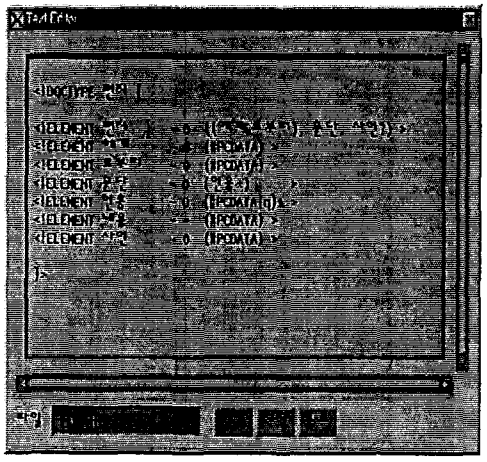
일반 사용자가 복잡한 DTD를 편집하기 위해서 메뉴를 이용하는 방법보다는 툴바상의 아이콘을 이용함으로써 보다 편리하고 빠른 작업 수행이 이루어질수록 하기 위해, WYSIWYG (What You See Is What You Get) 방식으로 결과를 보면서 작업을 수행할 수 있도록 하였다. 대부분의 작업이 마우스의 조작만으로 가능

하며 트리에는 엘리먼트, 엔티티, 속성 등만 화면에 보여주고 나머지 항목은 다른 윈도우에서 관리하게 되어 있어 화면의 복잡성을 줄였다.

Ⅳ. 구현 및 고찰

4.1. 구현

구현된 편집 시스템은 DTD 편집 시스템으로서의 기능 및 사용자 환경을 만족하도록 설계 및 구현하였다. 구현 환경으로서 서버를 위한 호스트로는 Sun O.S. 4.1.3을 탑재한 SPARC 워크스테이션과 586 PC를 이용하였고, X-Windows, C 언어, OSF/Motif를 사용하여 시스템을 구축하였으며, 화면 디자인은 OSF/UIL로 접근하였다. 본 시스템에서 구현 결과의 DTD 트리 구조 예를 그림 2에 보였다. 또한 트리 구조 형태로 작성된 DTD를 SGML 형태로 변환하여 저장한 뒤 미세 조정을 위해 본 시스템에 포함된 텍스트 편집기를 사용하여 수정하는 예를 그림 12에 보인다.



(그림 12) 텍스트 편집기

4.2. 고찰

이 기종 시스템간의 전자 문서의 효율적인 처리를 위해서는 문서의 개념적인 논리 구조를 표현하고 있는 SGML DTD를 일반 사용자도 쉽게 파악하고 편집할 수 있도록 구성되어야 한다. 그러나 일반적으로 DTD를 분석하거나 편집하기 위해서는 일반인이 접근하기가 쉽지 않았다.

이에, 본 논문에서는 대화적인 방법으로 SGML DTD를 편집할 수 있는 시스템을 구현 및 설계하였으며, 일반 사용자도 쉽게 문서의 구조를 이해하고 편집할 수 있도록 그래픽 환경하에서 트리 구조로 표현하였다. 따라서 SGML에 대해 전문적인 지식 없이도 간단히 DTD를 생성 및 조작할 수 있도록 설계되어 있고, DTD 생성 중 발생되는 엘리먼트와 엔티티 이외의 항목에 대해서는 별도의 윈도우를 사용해 관리하도록 하여 트리의 복잡성을 줄였다. 엘리먼트나 엔티티의 추가도 WYSIWYG 기능을 이용해 간단히 툴바의 아이콘을 사용하여 작업 시간을 단축하였다. 숨김 및 복구 기능도 트리의 복잡성을 줄이는데 사용되며, 주메뉴 상의 툴바를 이용한 DTD 생성은 능률적인 면에서나 시간적인 면에서 효과가 있을 것이라 생각된다. 공용화된 SP 파서를 이용하여 DTD 편집기나 텍스트 편집기를 사용해 파일로 저장할 때 이상 유무를 확인하므로 정확한 문서 구조를 얻을 수 있도록 하였다. 또한 엘리먼트 명과 엔티티 명에서 한글 처리가 가능하였다.

SGML을 이용한 응용 분야 즉, CALS, 데이터베이스, 자동차 및 항공 산업분야, 출판 등에서 다양하게 요구됨에 따라 이들에서 요구되는 문서 부류를 생성하기 위해 본 시스템은 유용하게 사용되리라 사료된다.

앞으로 DTD 편집기에 의해 생성된 파일을 데이타베이스화 하여 많은 사람이 공유할 수 있도록 하고, SGML 문서 편집 시스템 및 문서 포맷팅을 위해 DSSSL(Document Style Semantics and Specification Language) 및 SPDL(Standard Page Description Language) 등의 연구와 함께 시변수를 갖는 하이퍼미디어 문서 등을 위해 HyTime(Hypermedia/Timed-based Structuring Language) 등의 연구가 병행되어야 할 것이다.

다음에는 본 시스템과 기존의 시스템을 비교 설명한다. 본 시스템에서는 DTD를 그래픽 환경하에서 트리 형태로 표현함으로써 일반 사용자도 쉽게 DTD를 편집할 수 있으며, DTD 생성 시 필요한 항목은 메뉴 선택 방식에 의한 표현보다는 단순히 메뉴바의 아이콘을 선택해 추가할 항목에 마우스의 클릭만으로 모든 작업이 완료된다. 엘리먼트와 엔티티 이외의 속성 선언, 표기법 선언, 주석 선언 등은 별도의 윈도우를 두어 관리를 하였으며 숨김 및 복구 기능을 두어 트리의 복잡성을 최소화하였다.

표 1에서 본 시스템 이외의 다른 시스템들은 한글 태그를 사용할 수 없으나, 본 시스템은 한글 태그를 사용할 수 있으므로 영문 형식의 태그보다는 일반 사용자가 보다 쉽게 DTD 생성에 접근할 수 있는 장점이 있다.

이와 같이 본 시스템은 DTD에 대한 전문적인 지식이 없어도 간단히 DTD를 생성 및 편집할 수 있도록 트리 형태로 표현하였으며, 생성된 DTD는 SP 파서를 통해 자동적으로 오류를 검색할 수 있도록 설계하였다.

〈표 1〉 시스템 성능 비교

저작 도구	운영체제	특징 및 장점	단 점
DTD Editor	DOS	도스 환경 하에서 간단히 DTD를 생성	복잡한 DTD를 편집하기는 어려움
Rules Builder	Windows, Sun/SPARC, HP9000, MAC	쉽게 DTD를 생성, 간단한 오류도 built-in rules에 의해 기능적으로 체크, 다중 윈도우 지원.	지원되는 기능은 많으나 상용이다.
Inside & Out	MS Windows 3.1 (GUI)	구조화된 도표 방식 및 GUI를 사용하여 사용이 쉬우며 공개이다. 간단한 DTD를 생성시는 전혀 지장이 없다.	도움말 기능이 거의 없고, 미비한 기능과 메뉴도 많다.
Easy DTD	DOS/UNIX	2개의 공백 문자로 이루어진 indent를 이용한 계층구조와 DTD 규칙들을 사용하여 간단한 파일 입력으로 쉽게 DTD를 생성함.	생성된 DTD는 미비한 점이 많아 이를 수정하는데 많은 시간과 노력이 든다.
Near & Far	MS Windows 3.1 (GUI)	트리 구조 방식의 도표 사용, 트리 구조 방식의 도표를 사용하므로 문서의 구조 파악과 DTD 작성이 쉬우며 지원하는 기능도 다양하다.	지원되는 기능은 많으나 상용이다.
본 시스템	UNIX (GUI)	트리 구조 방식의 도표를 사용하여 문서의 구조 파악과 DTD 작성이 쉬우며, WYSIWYG 기능을 사용해 사용자의 작업을 편리하게 하며, 한글 태그를 처리할 수 있다. 엘리먼트, 엔티티 이외의 항목은 별도의 윈도우에서 관리함.	파서(parser)가 내장되어 있지 않으므로 저장하기 전까지는 간단한 오류 검색 이외에는 문법적 검색이 어려움

V. 결론

개방 환경에서 멀티미디어 정보를 이 기종 시스템간의 효율적인 교환을 위한 국제 표준인 SGML의 중요성은 이미 널리 확대되고 있으며, CALS의 도입으로 SGML 문서 처리 환경의 구축을 필요로 하는 시점에 있으나 대부분이 외국에 의존적이어서 많은 어려움을 겪고 있다. 특히 SGML은 장치에 독립적으로 문서의 개념적인 논리 구조를 가지고 있어 멀티미디어 문서 및 하이퍼미디어 문서도 수용할 수 있는 형식을 갖고 있다.

이에 따라, 본 논문에서는 SGML 문서 중 중요한 부분을 차지하고 있는 문서형 정의부(DTD)를 편집 및 생성할 수 있는 시스템에 대해 설계 및 구현하였으며, SGML 문법을 정확히 알지 못하는 사용자를 위해 WYSIWYG 기능을 활용하여 편의를 도모하였다. 또한 데이터를 트리 형태로 보여줌으로써 각각의 데이터에 대한 계층 구조를 쉽게 파악할 수 있게 하였고, 한글 처리가 가능하여 한글 태그를 갖는 DTD를 생성할 수 있다. 엘리먼트와 엔티티 이외의 부가적인 항목에 대해서는 별도의 윈도우를 두어 관리하였으며, 파일의 간단한 편집은 내장된 텍스트 편집기를 이용하였다.

본 시스템은 CALS, 데이터베이스, 자동차 및 항공 산업분야 등 많은 분야에서 널리 사용될 수 있으리라 사료되며, 향후 SGML의 보급과 실용화를 위해 SGML 편집기 및 SGML 파서, SGML 시스템 적합성 시험기, DTD 편집기에 의해 생성된 DTD와 데이타베이스와의 접목 등의 구현 및 설계에 관한 연구가 계속적으로 진행되어야 할 것이다. 그리고 SGML의 응용 분야인 HyTime과 SPDL, DSSSL 등 관련 표준에 대한 연구가 병행되어야 할 것이다.

참고문헌

1. 강치원, SGML DTD 작성을 위한 대화형 편집시스템에 관한 연구, 광운대학교 석사학위 논문, 1994
2. 김성혁 역, SGML의 기본과 이해, 성안당, 1996
3. 김충석, X 윈도우 시스템 프로그래밍, 이한출판사, 1994
4. 김경태, 민용식, 데이타 구조론, 정익사, 1990
5. 셈틀지기, C언어 사용자를 위한 MOTIF PROGRAMMING, 에스컴, 1994
6. 정회경, SGML & DTD, 제4 회 WWW 워크숍, 1996
7. 정회경, 문서처리 표준화 동향, 제2회 전산망 기술 및 표준화 심포지움, 1996
8. Martin Bryan. SGML, Addison-Wesley Publishing Company, 1991
9. Aaron M. Tenenbaum, Yedidyah Langsam, Moshe J. Augenstein, Data Structures Using C, PRENTICE HALL, 1990
10. Liora Alschuler, ABCD...SGML, INTERNATIONAL THOMSON COMPUTER PRESS, 1995
11. David Barron, Why use SGML, Eletronic Publishing, Vol.2(1), pp3-24, 1989
12. Donald D.Chamberlin and Charles F.Goldfarb, Graphic application of the Standard Generalized Markup Language,

Compute & Graphics, 1987

13. Eric van Herwijnen, PRACTICAL SGML SECOND EDITION, KLUWER ACADEMIC PUBLISHERS, 1994
14. Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, Fundamentals of Data Structures in C, COMPUTER SCIENCE PRESS, 1993
15. Ian A.Macleod, D.T. Barnard, D. Hamilton and M.Levison, SGML Documents and Non-linear Text Retrial, 1993
16. ISO TR 10037 Information Processing - Text and Office System-Guideline for SGML Syntax-Directed Editing Systems
17. J.Warmer and S. van Egmond, The Implementation of the Amsterdam SGML Parser, Eletronic Publishing, Origination Dissemination and Design, J.Wiley & Sons, Ltd., Chichester, July 1989

Design and Implementation of Interactive Editing System for SGML DTD Composition

Chang-Soo, Kim/Hoe-Kyung, Jung*

Abstract

This papar describes an design and implementation of a rule builder, named SGML DTD(Document Type Definition) Editor conforming to ISO 8879(SGML : Standard Generalized Markup Language).

SGML DTD define types of logical structure in documents and it is very hard to managing, operating with general common text editors because of it's complex structures. Therefore, We studied DTD automatic editor and production system, prototype system, direct operations on graphic trees in the environment of windows. It's easy to handle with general users. So, We analyzed the structures of document, and studied about SGML documents operation models. And also described logical structures by tree on Windows.

* Dept. Of Computer Engineering, Paichai Univ..