XML 어플리케이션을 위한 트리 기반 문서 편집 시스템의 설계 및 구현

김 영 철[†] · 강 춘 길[††]

요 약

본 논문에서는 구조 중심형 환경에서 사용 가능한 트리기반 XML 어플리케이션 문서 편집 시스템의 설계 및 구현에 관하여 논의한다. 본 시스템은 DTD를 ASTD(Abstract Syntax Tree Definition)로 변환하여 내장하는 구조로서 잘 설계된(well-formed) 문서는 물론 편집 중에 유효한(valid) 문서를 작성하도록 하기 위하여 구문 지향 편집(syntax-directed editing)을 지원한다. 또한 구조 중심형 환경에서 사용자와의 인터페이스를 담당하는 편집기를 설계함에 있어 새로운 도구의 추가로 인한 기능 확장이 용이하도록 고려하였으며, 편집기의 구현 시 기존의 문법 검증에 대한 비효율성을 감안하여 다중 엔트리(multiple entry)를 적용한 파서를 사용하여 실시간으로 유효성 검증이 가능하도록 하였다. 본 논문은 XML 문서 편집 시스템의 개발 모델을 제시함으로써 관련 어플리케이션 개발에 크게 기여할 것으로 기대된다.

A Design and Implementation of the Tree-based Document Editing System for XML Application

Young Chul Kim[†] · Chun Kil Kang[††]

ABSTRACT

This paper describes a design and implementation of the tree-based document editing system for XML application, available at the structure-oriented environment. This system converts DTD to ASTD(Abstract Syntax Tree Definition) to support syntax-directed editing for valid document, considers the extensibility to add new tools and supports multiple entry parser for real-time document validation. It is expected that this paper contributes related XML application document editing system development model.

키워드 : XML, 트리기반문서편집기(Tree-based Document Editor), DTD, 구문트리(AST)

1. 서 론

1.1 연구 배경 및 목적

XML(eXtensible Markup Language)은 HTML이 가지고 있는 단점들을 보완하면서, 인터넷에서 사용할 수 있는 문서로서의 규격을 갖추고 있다. 기존의 HTML 문서는 문서의 구조가 고정되어 있으므로 확장성이 없고, 문서의 유효성을 검사할 수 없다. 그러나 XML은 HTML이 갖는 문서 구조의 확장성과 문서의 검사기능에 대한 문제점을 극복하였다. 또한 태그의 최소화, 태그의 생략과 같이 SGML에서 처리가 난해했던 부분들을 제거함으로서 문서를 처리하기 쉽고, 명확하게 하였다. 이로 인해서 문서의 구조를 쉽게 파악할 수 있고, 문서의 검색이나 변환에 강한 장점을 가지고 있다. 특히 인터넷상에서 사용될 수 있도록 설계되었기 때문에 SGML에서 제공하기 어려운 URL을 사용하는 문서간의 링크를 HTML에서 사용하는 링크를 확장해서 더 강화했다. 따라서 XML 문서가 HTML문서를 대체할 수 있는 문서로 두각 되고 있고, 사용범위가 크게 확대될 것으로 기대된다[1, 2].

XML은 HTML이나 SGML과 그 특성이 상이함으로 기존의 파서나 브라우저를 활용할 수 없다. 그러므로 XML 문서를 적극적으로 활용하기 위해서 문서의 유효성을 검사하는 파서, 문서를 시각화해주는 브라우저, 문서의 생산성을 향상시키는 편집기가 절실히 요구되고 있다. 따라서 XML 어플리케이션을 위한 트리 기반 문서 편집기의 설계 및 구현은 단계별 XML 어플리케이션 개발 방법 제시로 문서 생산성에 크게 기여할 것으로 기대된다[3].

본 논문에서는 XML 어플리케이션을 위한 문서 편집 시스템에 관련된 기술을 두 가지로 제시하였다. 첫 번째는 XML 파서에서 문서의 구조를 파악하는 기술이고, 두 번째는 문서의 편집을 위해서 편집기에서 문서를 시각화하는 기술이다. 파서에서는 XML 어플리케이션 DTD를 가지고

† 준 회 원 : (주) 뉴스텍시스템즈 이사
†† 준 회 원 : (주) 엑스씨이 선임연구원
논문접수 : 2004년 3월 9일, 심사완료 : 2004년 5월 10일

문서 인스턴스(document instance)를 파싱함으로써 구문분석(syntax analysis)과 의미 분석(semantic analysis)이 이루어진다. 즉, 문서가 올바르게 작성되었는지와 유효한지를 검사하게 된다. 이 과정을 통해서 XML 어플리케이션 문서의 구조적인 정보를 얻어내고, 이를 모호하지 않은 형태의 자료구조에 저장한다. 편집기에서는 XML 어플리케이션 문서를 시각화하기 위해서 XML AST 정의의 역파싱 규칙(unparsing scheme)을 이용하는데, 이를 위해서 XML 문서를 처리해서 생성된 문서 인스턴스 트리 정보와 역파싱 규칙을 매핑시켜 화면에 출력하도록 하였다. 문서 편집 시에는 문서 시각화와 시각 프로그래밍을 동시에 제공하여 각각이 지니는 단점을 상호보완하고, 개발자에게는 일관된 시각기호를 제공하므로써 개발 대상에 대한 인지도를 높이고자 한다. 이러한 시스템에서 갖추어야할 조건으로는 먼저 사용하는 시각기호가 문서 시각화와 시각 프로그래밍 모두에 적절해야 한다는 점과 시스템에 생길 확장 등을 고려하여 시스템이 유연하게 설계되어야 한다는 점을 들 수 있다.

본 논문은 다음과 같은 형태로 구성되어 있다. 제 2장에서는 XML 편집 시스템에 대한 관련 연구를 제시하였으며, 제 3장에서는 XML 편집 시스템의 설계에 대해서 기술하였다. 또한 제 4장과 제 5장에서는 각각 구현과 결론 및 향후 연구에 대해서 기술하였다.

2. 관련 연구

XML을 위한 편집 시스템은 현재 여러 가지 언어로 제작되고 있는데, DTD를 가지고 유효성 검사(validating)을 하는 편집기와 DTD없이 잘 설계되었는지(well-formedness)만을 검사하는 편집기로 나뉘어 진다. 또한 현재의 편집기들은 텍스트 편집 환경보다는 트리와 같은 그래픽 사용자 인터페이스를 사용해서 보다 직관적으로 편집이 가능하다. 그러나 이들은 XML 문서처리를 완벽하게 처리하지 못하거나, XML의 변형된 형태의 문서를 처리하거나, SGML에 관련된 소프트웨어를 수정한 형태를 가짐으로서 순수한 XML 문서를 처리하는데 적합하지 못하며, XML 어플리케이션 사용자를 위한 소프트웨어로 부족한 점이 많다. 현재 특정 XML 어플리케이션을 위한 편집기는 찾아보기 힘든 실정이다.

Techno2000에서 개발한 CLIP! XML Editor[4]는 트리 기반, 텍스트 기반의 편집을 지원하며 새문서 작성 마법사를 통해서 처음부터 마지막 단계까지 단계별로 가이드 해주는 마법사를 이용한 문서 저작을 지원한다. 또한 DTD 트리보기 기능, 에러 출력 및 수정 기능을 제공하며 잘 설계된(well-formed) XML 문서에서 DTD를 추출, 생성해 준다. 생성된 DTD는 비슷한 구조를 갖는 문서의 저작에 이용할 수 있어 다수의 문서를 쉽게 작성할 수 있다. 기본적인 텍스트 검색과 엘리먼트 검색 외에도 컨텐트에 기반한 엘리먼트 검색과 컴포넌트 검색 등 확장 검색이 가능해 원하는 정보를 쉽게 찾을 수 있다. 오류 수정 기능으로는 문서의 유효성 검사 시 옵션에 의해 첫 번째 오류 감지에서 검사를 멈출 수도 있고 문서전체의 오류를 출력하게 할 수도 있다. 출력된 오류 메시지를 선택하면 오류가 발생한 곳으로 바로 이동할 수 있어 쉽게 오류를 수정할 수 있다. 마지막으로 다른 XML 문서들의 엘리먼트를 불러와 재사용할 수 있어 다량의 XML 문서 저작 시 문서 작성 시간을 최소화할 수 있다.

Vervet Logic에서 개발한 XML Pro v2.01[5]은 순수 자바 애플리케이션으로 쉽고 직관적인 그래픽 사용자 인터페이스, 문서 구조를 유지하는 문서 트리 편집 뷰, DTD를 통한 XML 유효화 검증을 지원한다. 또한 편리한 엘리먼트 생성 및 관리를 위한 엘리먼트 마법사(element wizard), 속성 생성 및 관리를 위한 속성 마법사(attribute wizard)도 제공된다. 기본적으로 엔티티(entity), CDATA, 주석을 포함한 W3C의 XML 1.0 스펙을 지원한다.

Cuesoft의 EXml[6]은 well-formed XML 문서 편집기로서 트리나 소스 뷰를 지원한다. 트리 뷰를 통해서 엘리먼트 편집이 가능하다. 다른 편집기와 마찬가지로 well-formedness를 체크하는 것 이외에 특이한 사항은 XSL 이름공간(namespace)지원과 PCDATA, 주석, 속성값을 엘리먼트에 매핑한 테이블 출력, 소스뷰 상에서의 직접접적인 텍스트 편집이 가능하다는 것이다.

Altova의 xmlspy® 2004[7]는 유효한 XML 문서를 쉽게 만들 수 있도록 템플릿과 상황에 대응하는(context-sensitive) 편집, 필요한 엘리먼트와 속성을 자동으로 채우는 기능(auto-completion)을 지원한다. 또한 현재 위치에서 어떤 엘리먼트를 사용할 수 있는지 알려주는 엔트리 헬퍼(Entry Helper)를 지원을 통한 직관적인 사용자 인터페이스를 제공하고 DTD 편집기, XSL 편집 및 디버깅 기능 등을 지원한다.

3. 트리기반 문서 시스템 설계

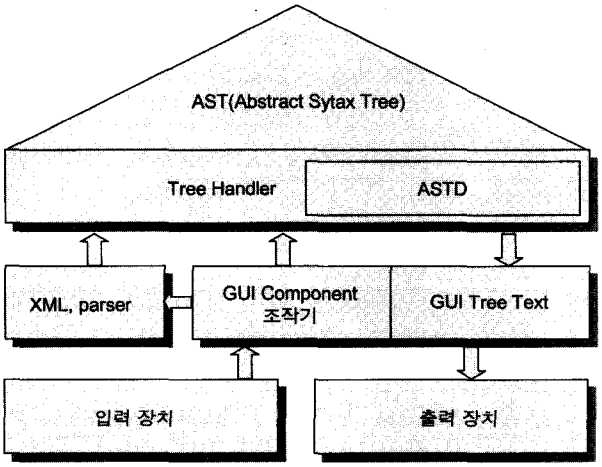
본 논문에서 제시하고자 하는 트리 기반 문서 편집 시스템은 XML 어플리케이션을 위한 개발환경으로 구현되었으며, XML 문서의 시각화에 초점이 맞추어져 있다.

3.1 시스템 구조

다음 (그림 1)은 본 논문에서 설계한 시스템 모델이다. (그림 1)에서 보는바와 같이 본 논문에서 구현한 XML 편집기는 AST, ASTD, 파서, 인터프리터, 역파서로 이루어졌다.

ASTD(Abstract Syntax Tree Definition)는 DTD를 내부 편집기나 역파서에서 사용할 수 있도록 만들어놓은 문법 트리이다. 편집기에서 만들어지는 모든 문서는 ASTD에 의해서 문법이 검증되어 올바른 문서만 만들어진다. 또한 AST는 ASTD를 통한 유효성 검증을 거쳐서 만든 내부 자

료구조이다. 파서(Parser)는 문서 유효성 검증을 한다. 파서는 문서의 토큰을 인식해 내는 어휘 분석기와 일련의 토큰의 조합이 적합한 구문 구조를 갖는가를 검사하는 구문 분석기로 구성된다. 또한 역파서(Unparser)는 AST의 각 노드를 그래픽 트리 구조로 보여주는 역할을 한다. 이것은 일반 사용자들이 XML을 직접 편집할 수가 없고, 대신에 그래픽 사용자 인터페이스 편집 환경을 통해서 편집해 나가기 때문에 필요하다. 역파서는 AST를 그래픽 트리 구조로 역파싱하는 기능 외에 일반 텍스트 문서로 저장도 한다. 트리 조작기(Tree Handler)는 ASTD를 참조해서 실제로 문법이 올바른 AST 노드를 추가/삭제/갱신/변환에 대한 제어를 한다. 명령어 처리기(Command Interpreter)는 사용자가 편집 환경에서 입력한 명령어를 해석하는 부분으로 입력된 명령어가 수행 가능한 것인지를 검사하고 수행 가능한 명령어인 경우에는 해당 기능을 수행한다. 이때 각각의 명령은 트리 조작기를 통해서 실제로 내부 트리 연산이 발생하게 된다.



(그림 1) XML 트리기반 문서 편집기 모델

3.2 ASTD(Abstract Syntax Tree Definition)

편집기의 핵심은 DTD를 문법 규칙의 집합인 추상 문법(abstract syntax)으로 정의하는 것이다. 편집하는 개체(object)는 문법에 따라서 유도 트리(deprivation tree), 즉 AST로 표현된다. 편집기 사용자 인터페이스의 조작에 따른 텍스트나 문서 구조의 변화는 주어진 구문 트리가 변화함을 의미한다. 본 논문에서는 XML 어플리케이션 중에서 VoiceXML의 DTD를 ASTD로 변환하였다. 추상 문법은 다음과 같은 형식의 프로덕션(production)의 집합으로 이루어진다[6].

$$x_0 : op(x_1, x_2 \cdots x_k) ;$$

여기서 op는 연산자명(operator name)이고 각 x_i는 문법의 비단말명(nonterminal name)이다. 프로덕션 인스턴스를 구별하기 위한 목적인 연산자를 제외하고 문법 규칙은 다음과 같이 문맥 자유 프로덕션(context-free production)으로 이루어졌다.

$$x_0 \rightarrow x_1 x_2 \cdots x_k$$

다음의 <표 1>은 XML 어플리케이션의 하나인 VoiceXML DTD의 일부를 ASTD로 변환한 예를 보여준다.

〈표 1〉 DTD의 ASTD변환

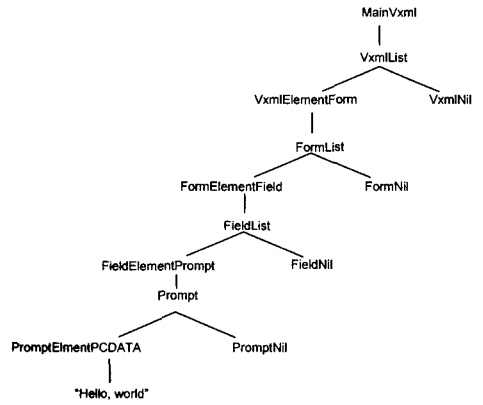
<VoiceXML DTD>	<변환된 ASTD>
	main : MainVxml(vxml) ; vxml : VxmlNil( ) | VxmlList (vxmlElement vxml) ;
	vxmlElement : VxmlElelementNil () | VxmlElementCatch (catcch) | VxmlElementHelp (help) | VxmlElementNoinput (noinput) | VxmlElementNomatch (nomatch) | VxmlElementError (error) | VxmlElementForm (form) | VxmlElementMenu (menu) | VxmlElementLink (link) | VxmlElementMeta (meta) | VxmlElementProperty (property) | VxmlElementVar (var) | VxmlElementScript (script) | VxmlElementComment (comment) ;
<!ELEMENT vxml (catch | help | noinput |nomatch | error | form | link |menu | meta | property | script | var)+>	
<!ELEMENT form (grammar | initial | subdialog | field | filled | record | dtmf | block | var | catch | help | noinput | nomatch | error | object | link | transfer | property | comment)* >	formElement : FormElementNil() | FormElementGrammar (grammar) | FormElementInitial (initial) | FormElementSubdialog (subdialog) | FormElementField (field) | FormElementFilled (filled) | FormElementRecord (record) | FormElementDtmf (dtmf) | FormElementBlock (block) | FormElementVar (var) | FormElementCatch (catch) | FormElementHelp (help) | FormElementNoinput (noinput) | FormElementNomatch (nomatch) | FormElementError (error) | FormElementObject (object) | FormElementLink (link) | FormElementTransfer (transfer) | FormElementProperty (property) | FormElementComment (comment) ;

3.3 추상 구문 트리(Abstract Syntax Tree)

AST는 XML 문서를 파싱하여 편집기에서 편집 가능하도록 설계한 자료구조이다. 사용자가 편집기에서 행하는 편집 연산에 대한 처리는 모두 AST 인스턴스에 반영된다. 다음 (그림 2)는 앞서 정의한 ASTD를 통해서 <표 2>의 XML 문서를 AST로 나타내주는 예를 보여준다. 이 AST는 내부적으로 제어가 가능한 구조이므로 명령어 처리기에서 전달한 현재 명령 이외에 추가적인 명령에 대해서 구현이 가능하다.

〈표 2〉 XML 문서

```
< ?xml version = "1.0" >
< !DOCTYPE vxml SYSTEM "vxml.dtd" >
< vxml version = "1.0">
    < form >
        < field >
            < prompt >
                Hello, world
            < /prompt >
        < /field>
    < /form>
< /vxml>
```



(그림 2) XML문서의 AST 인스턴스

AST를 구성하는 노드의 구조는 <표 3>과 같다. 이 노드는 문서 내에서 사용되는 태그 이름과 연관된 속성정보와 AST에서 다른 노드들과의 연결 정보들로 구성된다. 문자열 데이터는 작성자가 직접 작성한 문서 내용을 갖고 있는 부분으로, 이것은 PCDATA, CDATA나 주석과 같은 서브 엘리먼트로 이루어진 엘리먼트인 경우에만 유효한 정보이다. 실제 이 부분은 문자열만을 저장한다.

〈표 3〉 AST 노드 자료구조

생성 번호				
문자열 데이터				
속성 리스트				
부모 노드 연결	오른쪽 노드 연결	왼쪽 노드 연결	다음 노드 연결	이전 노드 연결

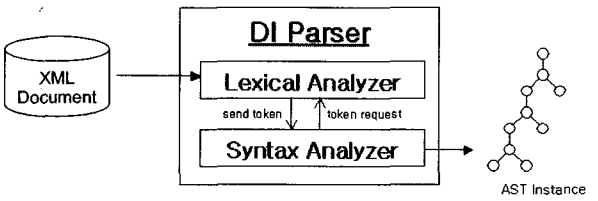
AST 노드의 연결정보는 5가지로 구분한다. 먼저 물리적인 노드의 연결 정보로 부모, 오른쪽, 왼쪽 노드 연결 정보가 있다. 문서의 AST는 편집기에서 항상 이용되는 문서 정보이므로, 편집기에서 작성자가 선택할 수 있는 문서 태그 위치에 제한을 두지 않게 하기 위하여 이전 노드와 다음 노드 연결 정보를 추가하였다. 이전 노드 연결과 다음 노드 연결의 기본적인 트리 순회방식은 프리오더(preorder) 방식을 이용한다.

3.4 XML 어플리케이션 DI 파서와 역파서

XML 문서는 실제로 작성되는 문서 내용으로서 DI(Document Instance)라고 한다. 특히 이 DI는 XML DTD에 따라 종속적으로 작성되는 문서이므로 사용자는 DTD를 숙지해야만 하는 번거로움이 있다. 따라서 특정 DTD를 적용한 XML 어플리케이션 편집기를 사용하면 사용자의 부담이 줄어준다. 본 논문에서 처리하는 트리 형태의 XML 문서 구조를 AST 인스턴스(AST instance)라고 한다. DI 파서는 XML 어플리케이션 문서를 파싱하여 AST 인스턴스를 생성한다.

3.4.1 파서의 구성

파서의 구성은 (그림 3)과 같이 크게 2가지 부분으로서 어휘 분석을 담당하는 어휘 분석기와 구문 분석을 담당하는 구문분석기로 구성된다[9].

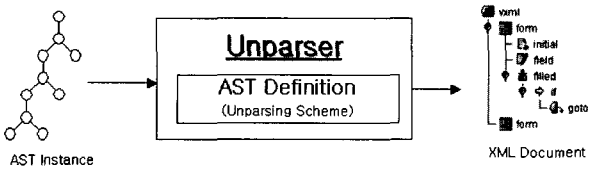


(그림 3) 문서 인스턴스 파서

어휘 분석기(Lexical Analyzer)는 DI에서 일련의 문자를 받아서 토큰으로 인식하여 해당 토큰을 구문 분석기에 전달한다. 토큰은 각 엘리먼트의 태그와 속성 및 주석으로 구성되어 있다. 구문 분석기(Syntax Analyzer)는 어휘 분석기로부터 토큰을 받아서 주어진 토큰의 조합이 문법에 올바른 것인지를 검증한다. 이 때 하향식 파싱(top-down parsing)의 대표적인 방식이며 다중 엔트리(multiple entry)를 지원하는 재귀적 자동 파싱(recursive descent parsing : RDP) 기법을 사용한다. RDP는 결정적 파서(deterministic parser)로서 비결정적 파서(non-deterministic parser) 보다 빠르며, 다중 엔트리를 통해서 문서 일부에 대한 유효성 검증이 가능하다.

3.4.2 역파서(Unparser)

역파서는 (그림 4)와 같이 AST 인스턴스를 입력으로 받아들이고 원래 문서 형태를 ASTD의 역파싱 규칙(unparsing rule)을 참조하여 출력장치(화면 혹은 디스크)에 적당한 형태를 갖추어 보기 좋게 출력한다.



(그림 4) 역파서

역파싱 규칙은 다음과 같다.

Phylum : operator [selection symbols]

여기서 꺾쇠괄호(square bracket)는 해당 프로덕션의 이미지이다. 선택 심볼(selection symbol)은 @과 이름으로 나누어지는데 @은 자식 프로덕션을 역파싱 함을 의미하고, 이름은 트리 상에서 출력할 아이콘이나 텍스트를 결정한다. 실제로 ASTD와 AST 인스턴스를 이용하여 역파싱을 하는 역파싱 알고리즘은 <표 4>와 같다.

〈표 4〉 역파싱 알고리즘

```
Algorithm UNPARSING( P, N)
Input : Node
Output : 역파싱된 출력물
Definition :
    P = 부모 노드
    N = 현재 노드
Method :
    if( N이 리스트 노드이면) {
        do {
            AST정의에서 N의 프로덕션에 대한 역파싱 규칙을 읽는다.

            if( 읽은 역파싱 규칙이 첫 번째 @이면)
                UNPARSING( N, N의 왼쪽 노드 링크)
            else if( 읽은 역파싱 규칙이 두 번째 @이면)
                UNPARSING( N, N의 왼쪽 노드 링크)
        } while( 역파싱 규칙이 남아 있을 경우)
    }
    else {
        do {
            AST정의에서 N의 프로덕션에 대한 역파싱 규칙을 읽는다.

            if( AST정의에서 N의 프로덕션 메뉴값이 STRING이면)
                P의 자식으로 STRING을 출력한다.
            else if( 읽은 역파싱 규칙이 @이면) {
                if( N의 Prod에 대한 메뉴값이 STRING이면)
                    P의 자식으로 N의 왼쪽 노드 링크를 출력한다.
                else
                    UNPARSING( N, N의 왼쪽 링크 노드)
            }
        } while( 언파싱 규칙이 남아 있을 경우)
    }
```

역파싱 알고리즘은 처음에 노드가 리스트 노드인가 아닌가를 검사한다. 노드가 리스트인 경우에는 화면에 출력이 없고 자식 노드에 대한 역파싱 루틴 호출만 있다. 역파싱 규칙이 첫 번째 @인 경우에는 왼쪽 노드 링크를, 두 번째 @인 경우에는 오른쪽 노드 링크를 현재 노드와 함께 UNPARSING인자로 전달한다. 이 과정을 역파싱 규칙이 없을 때까지 진행한다. 노드가 리스트가 아닌 경우에는 @에 대한 처리 외에 화면 출력을 위한 실제적인 부분이 있다. 해당 노드의 프로덕션에 해당하는 역파싱 규칙을 읽은 후에 AST 정의의 메뉴값이 STRING인 경우는 주석이나 PCDATA인 경우로서 화면에 해당 노드를 만든 후에 실제 주석값이나 PCDATA값을 출력한다. STRING이 아닌 경우는 @으로서 이때는 오직 왼쪽 노드 링크만이 존재하므로 현재 노드와 함께 UNPARSING인자로 전달한다.

3.5 트리 조작기와 명령어 처리기

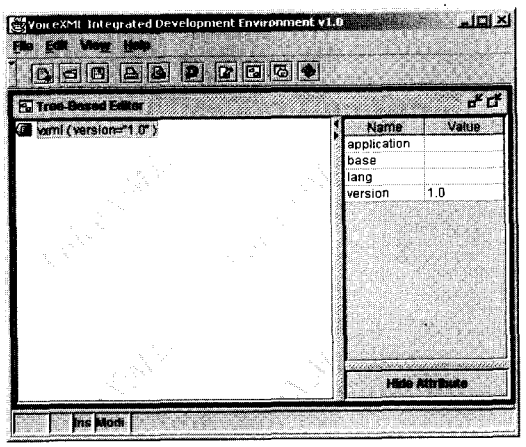
트리 조작기는 AST 인스턴스에 대한 직접적인 제어를 담당한다. 이 때 내부 트리를 방문하여 특정 노드를 생성/삭제함은 물론 ASTD를 사용하여 특정 노드에서 확장 가능한 엘리먼트를 찾아낼 수도 있다. 명령어 처리기는 사용자의 입력을 번역하여 일련의 연산을 트리 조작기에 전달한다. 이때 트리 조작기의 엘리먼트 단위 연산만을 사용한다.

4. 구 현

본 시스템에서는 XML 문서의 구조적 시각화를 위해서 자바 스윙(JDK 1.2.2에서 구현)에서 제공하는 트리와 테이블 인터페이스를 제공한다.

4.1 초기 화면

(그림 5)는 트리 기반 XML 어플리케이션 문서 편집기의 초기 새문서로 시작한 그림이다. 윈도우의 좌측은 엘리먼트 트리 윈도우로서(element tree window) XML 문서를 트리 구조로 보여주는 윈도우이다. 문서 작업의 대부분은 엘리먼트 트리 윈도우의 엘리먼트 편집으로 이루어진다. 우측은 속성 윈도우(attribute window)로서 엘리먼트 트리 윈도우에서 엘리먼트를 선택했을 때 해당 엘리먼트의 속성을 편집하는 윈도우이다. 또한 속성 윈도우 밑에는 속성 보이기/숨기기 버튼이 있다. 이 버튼은 좌측의 엘리먼트 트리 윈도우의 각 엘리먼트 노드에 속성 정보를 보이거나 숨기는 기능을 토글 시킨다. 문서의 구조적인 정보는 숨기기 설정하고 세부적인 내용은 보이기로 하는 설정하는 것이 좋다.

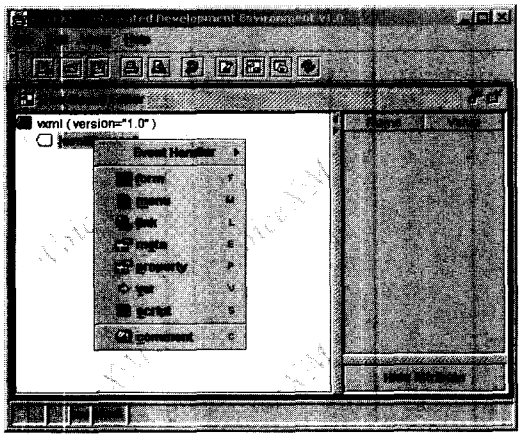


(그림 5) 초기 화면

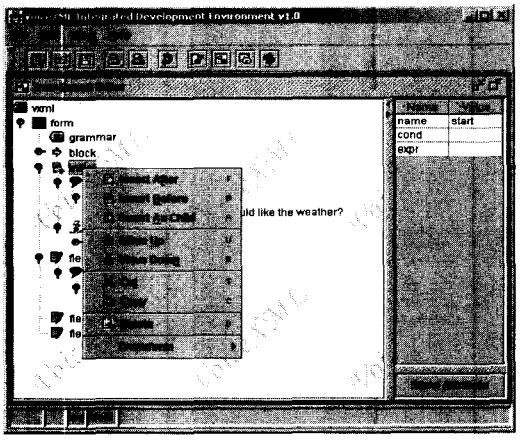
4.2 Transient Placeholder를 위한 인터페이스

엘리먼트 삽입에 관련된 연산(Insert After, Insert Before, Insert As Child)을 수행하면 (그림 6)와 같이 꺾쇠묶음([vxml

Element])으로 표현된 노드가 생성된다. Transient placeholder는 최종적으로 엘리먼트가 생성되기 위한 중간과정의 노드이다. 엘리먼트 생성을 위해서는 노드에 마우스 왼쪽 버튼을 누르면 (그림 6)과 같이 확장 가능한 엘리먼트가 팝업메뉴로 나온다. 엘리먼트의 선택은 placeholder를 생성한 후에나 가능하다. 이 때 마우스 왼쪽 버튼을 사용한다. 반면에 엘리먼트 편집에 관련된 연산은 모두 마우스 오른쪽 버튼을 사용한다. Placeholder를 포함한 모든 노드에 마우스 오른쪽 버튼을 누르면 (그림 7)과 같은 엘리먼트 편집 팝업 메뉴가 나온다.



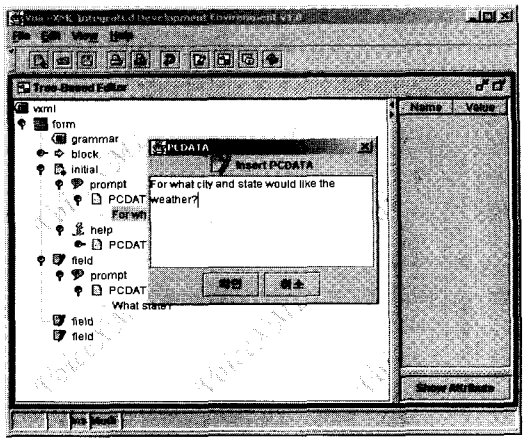
(그림 6) 확장 가능한 엘리먼트 팝업 메뉴



(그림 7) 엘리먼트 편집 팝업 메뉴

4.3 PCDATA, 주석 편집을 위한 인터페이스

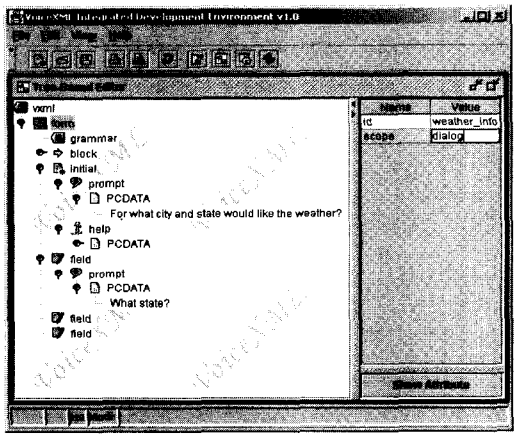
Placeholder를 포함한 모든 엘리먼트는 트리 윈도우에서 편집을 한다. 그러나 텍스트입력을 받는 PCDATA와 주석은 (그림 8)과 같이 따로 입력 대화상자를 가지고 편집을 한다. 이때 사용자가 입력을 마치고 확인 버튼을 눌렀을 때, 입력한 내용이 유효한지를 검사하기 위해서 내부 다중 엔트리 파서를 사용한다. 입력에 오류가 있을 경우에는 다시 입력을 받는다.



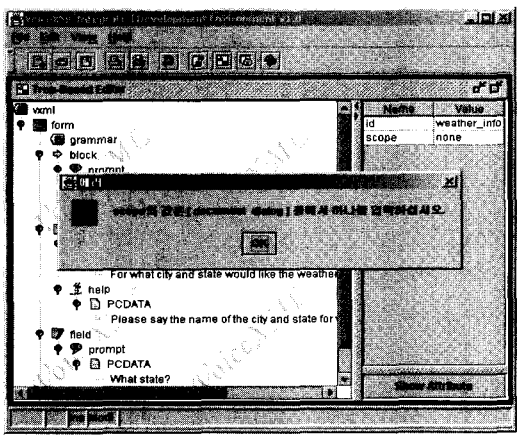
(그림 8) PCDATA 편집

4.4 속성 편집을 위한 인터페이스

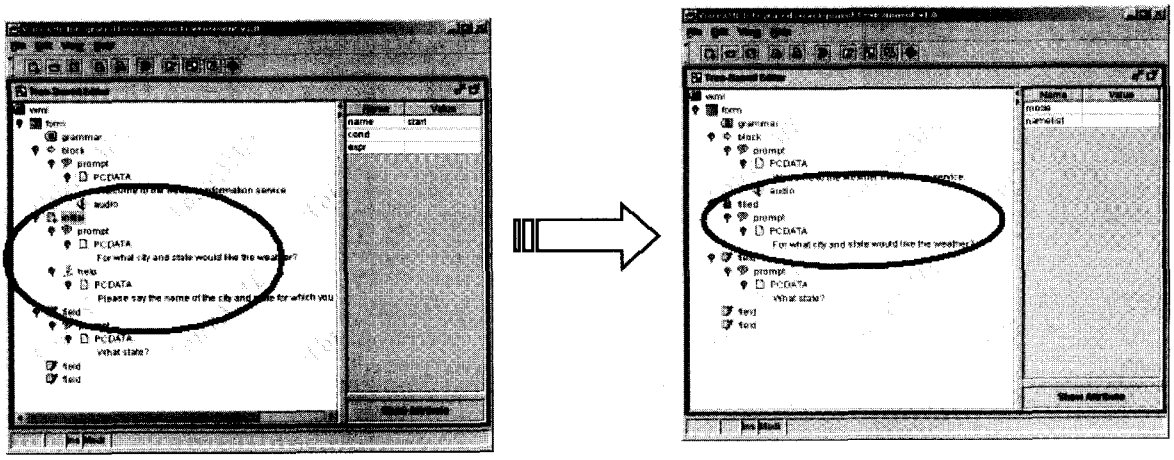
엘리먼트 트리 윈도우의 각 엘리먼트를 클릭 하면 이에 오른쪽 속성 윈도우에 해당 엘리먼트에 대한 속성명과 속성 값이 나온다. 이 때 각 값에 대한 편집이 가능하다. 다음 (그림 9)는 속성 편집 창을 보여준다. 각각의 속성 값에 대한 입력을 마친 후에는 AST정의의 속성 정보를 이용하



(그림 9) 속성 편집



(그림 10) 속성 편집 오류



(그림 11) 변환 엘리먼트

여 속성 값 유효성 검증을 한다. 만일 값에 오류가 발생한 경우에는 (그림 10)과 같은 오류 메시지가 나온다.

4.5 변환(Transform)을 위한 인터페이스

변환(Transform)을 위한 인터페이스는 다음 (그림 11)과 같이 특정 엘리먼트를 다른 엘리먼트로 변경하고자 할 때 이용하는 인터페이스이다[10, 11]. 변환을 사용하면 ASTD의 엘리먼트 정보를 참조하여 변경하고자 하는 엘리먼트와 현재 엘리먼트의 서브 엘리먼트 정보를 비교하여 문법에 어긋나는 엘리먼트는 지우는 동작을 취한다. 이렇게 함으로써 기존의 엘리먼트 정보에 대한 재사용이 가능하게 된다.

5. 결 론

본 논문에서는 XML 어플리케이션에 대한 편집 도구로서 기존의 수동으로 유효성 검사를 해야하는 방식의 불편함을 해소하고 일반 사용자가 쉽게 XML 문서를 편집할 수 있도록 구문 지향 방식을 적용한 XML 편집 시스템을 설계하고 구현하였다.

시스템 구현 시 XML 어플리케이션 DTD를 ASTD로 변환하고 재귀적파서(recursive descent parser)를 구현했다. ASTD의 역파싱 규칙을 통해서 AST 인스턴스를 화면이나 파일로 출력을 했으며, 문서 편집의 효율성과 문서 구조의 파악이 용이하게 하기 위하여 내부 문법이 익숙하지 않더라도 쉽게 사용이 가능하도록 문서 편집을 할 때 확장/치환 가능한 엘리먼트를 ASTD 참조를 통해 제시하는 언어-기반(language-based) 시스템[11]과 트리 기반의 비주얼 프로그래밍 환경을 제공하였다. 이 때 DTD를 ASTD와 다중 엔트리를 지원하는 파서를 내장함으로써 사용자들이 문서 편집 시 문법에 맞는 문서를 작성할 수 있도록 함으로써 문서의 오류를 줄였다. 따라서 사용자는 문서 작성 중 항상 유효(valid)한 문서만을 작성한다.

향후 연구과제는 DTD의 ASTD 변환을 자동화하고 이를 이용하여 실시간 유효성 검증이 가능한 범용 XML 편집기의 설계 및 구현이 이루어져야 할 것이다.

참 고 문 헌

[1] Extensible Markup Language(XML) 1.0, Avalilable http://www.w3.org/TR/ REC-xml, 1998.

[2] VoiceXML Forum, Voice eXtensible Markup Language 1.0, March, 2000.

[3] William J. Pardi. XML in Action : Web Technology (Microsoft Press), pp.31-71, pp.137-159, 1999.

[4] CLIP! XML Editor Avalilable http://xml.t2000.co.kr.

[5] XML Pro v2.0 Available http://www.vervet.com/products.php

[6] EXml v1.2 Available http://www.cuesoft.com/products/exml.asp.

[7] xmlspy® 2004, Available http://www.xmlspy.com/products_ide.html.

[8] Thosmas W. Reps, Tim Teitelbaum, The Synthesizer Generator : A System for Constructing Language-Based Editors, Springer-Verlag 1988.

[9] Aho, A., Sethi, R. and Ullman, J., Compilers : Principles, Techniques and Tools, Addison-Wesley, 1996.

[10] A. D. N. Edwards. Visual Programming Langauges : the Next Generation?. ACM SIGPLAN Notices, Vol. 23, No.4, pp.43-50.

[11] R. A. Ballance, J. Butcher and S. L. Graham, Grammatical abstraction and incremental syntax analysis in a language-based editor, Proceedings of the SIGPLAN '88 conference on Programming Language design and Implementation, pp.185-198, 1988.

김 영 철

e-mail : yckim@ss.ssu.ac.kr
1990년 한남대학교 전자계산학과(학사)
1996년 숭실대학교 대학원 전자계산학과 (공학석사)
2003년 숭실대학교 대학원 컴퓨터학과 (공학박사)
현재 (주) 뉴스텍시스템즈 이사, 명지전문대학 겸임조교수
관심분야 : 프로그래밍 언어, 컴파일러, 망관리, XML

강 춘 길

e-mail : gilbird@netian.com
1999년 숭실대학교 컴퓨터학부 학사
2001년 숭실대학교 컴퓨터학과 석사
현재 (주) 엑스씨이 선임연구원
관심분야 : 컴파일러, 시각 프로그래밍, XML, J2ME