HOMT : HyTime DTD 설계를 지원하기 위한 XOMT의 확장

장 원 호[†] · 임 혜 정[††] · 박 인 호[†] · 강 현 석[†††]

요 약

하이퍼미디어 응용들은 다양한 미디어들을 동적이고 비 순차적으로 연결하고 동기화 등의 시간성을 지원해야 한다. 최근 이러한 하이퍼미디어 문서를 표현하는 메타 언어로 HyTime(Hypermedia/Time based Structuring Language)이 사용되고 있다. 그런데 HyTime을 이용하여 하이퍼미디어 문서를 기술하고 관리하는 일은 비전문가들에게 상당히 어렵다.

본 논문은 하이퍼미디어 문서의 구조를 기술하는 HyTime DTD를 쉽게 설계할 수 있게 하는 다이아그래밍 기법인 HOMT(HyTime support XOMT)를 제안한다. HOMT는 SGML DTD를 설계하기 위해 제안된 XOMT(eXtended OMT)를 확장하는 방법으로 고안되었다.

HOMT : Enhancing XOMT to Support HyTime DTD Design

Won-Ho Jang[†] · Hye-Jung Lim[††] · In-Ho Park[†] · Hyun-Syug Kang[†††]

ABSTRACT

Hypermedia applications have to support a various media with time related functions that are linked with non-continuous and synchronous. Recently, HyTime(Hypermedia/Time based Structuring Language) can be used for this purpose. However, it is very difficult to describe and manage hypermedia documents using HyTime by non-specialists.

In this thesis, we propose a HOMT(HyTime support XOMT) scheme which is very easy to design HyTime DTD to describe the structure of hypermedia documents. HOMT scheme is contrived by extended method of XOMT(eXtended OMT) for design of SGML DTD.

1. 서 론

최근 WWW, CAD/CAM, CAI, CALS, OIS, AI, Digital Library 등의 새로운 컴퓨터 응용 분야에서 다양한 데이타를 효과적으로 관리할 필요가 생겼다. 그런데 이러한 응용들에서 나타나는 멀티미디어 데이타들은 종류가 다양하고 그들 사이의 관련도가 매우 복잡하기 때문에 종래의 상용 데이타 처리 방식으로는 효과적으로 관리하기가 어렵다. 따라서 기존의 관계형 데이타베이스 시스템들이 가진 문제점들을 극복하기 위해 객체 식별성, 클래스 상속, 캡슐화 등의 객체지향 개념들을 효과적으로 지원하는 객체지향 데이타베이스 시스템들이 유용성을 인정받고 있다[1, 2].

멀티미디어 데이타를 객체지향 데이타베이스에서 관리하기 위해서는 우선 체계적인 개발 방법론에 기초

※ 이 논문은 1997년도 정보통신부 대학기초연구지원사업의 지원을 받아 연구되었음.
† 정 회 원 : 경상대학교 컴퓨터과학과/전산개발연구소
†† 준 회 원 : 경상대학교 컴퓨터과학과/전산개발연구소
††† 종신회원 : 경상대학교 컴퓨터과학과/전산개발연구소 교수
논문접수 : 1997년 11월 17일, 심사완료 : 1998년 6월 29일

한 객체지향 데이타베이스 설계 도구가 지원되어야 한다. 특히 멀티미디어 데이타들로 구성되는 문서들을 데이타베이스로 관리하기 위해서는 멀티미디어 표준 문서를 나눌 수 있는 객체지향 개발 방법론이 정립되어야 하며, 이를 지원하는 설계 도구가 개발되어야 한다[3].

XOMT(eXtended OMT)[3, 4]는 SGML로 기술된 문서들을 효과적으로 객체지향 데이타베이스에서 다룰 수 있도록 다양한 그래픽 기능을 제공하고 설계 정보를 통합적으로 관리하고 있다. 하지만 링크에 따라 동적이고 비 순차적으로 수행되는 항행과 동기화 등의 시간성을 지원해야 하는 하이퍼미디어 응용들의 경우 다양한 미디어들과 그들 사이의 관련성을 효과적으로 관리하기가 더욱 어려우므로 통합적이고 체계적인 하이퍼미디어의 정보 관리를 위해 XOMT의 기능이 확장될 필요가 있다.

본 논문은 시간성을 지원하는 하이퍼미디어 문서를 표현하는 메타 언어인 HyTime(Hypermedia/Time based Structuring Language, ISO 10744)[5]을 이용하여 하이퍼미디어 문서를 효과적으로 기술할 수 있도록 SGML에 기반을 둔 XOMT를 확장한 HyTime 지원 시각적 객체 다이아그래밍 기법 HOMT(HyTime support XOMT)를 제안한다. 이는 기본적으로 SG ML DTD를 지원하는 XOMT의 기능을 모두 사용할 뿐만 아니라 HyTime의 6개 모듈들을 용이하게 Hy-Time DTD 설계에 사용할 수 있도록 한다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 논하고 2장은 SGML과 HyTime에 대해 간략하게 요약하고 HyTime의 지원을 위한 SGML의 확장에 대해 설명한다. 4장은 XOMT와 HyTime 지원을 위해 이를 확장한 새로운 다이아그래밍 기법인 HOMT를 제시한다. 5장에서는 HOMT 도구의 실제 적용 예를, 6장에서 결론 및 향후 과제에 대해 논의한다.

2. 관련 연구

HyTime은 SGML을 기반으로 하는 언어로서 일반 사용자도 쉽게 DTD를 설계할 수 있는 시각적인 객체 다이아그래밍 기법과 이를 지원하는 도구가 요구된다. 그러나 DTD 설계를 위한 대부분의 기존 도구들은 에디터 중심이다[13].

SGML의 DTD 설계를 위해 제안된 기존의 다이아그래밍 기법으로는 Herwijnen의 구조도(Structure Diagrams)가 있다. 이 방법은 DTD를 일반적인 트리 구조로 나타낸 것으로 각 노드를 왼쪽에서 오른쪽으로 놓아놓은 흐름도 형식이다. 그러나 간단한 DTD 표현은 편리하나 복잡한 엘리먼트의 표현이나 공유되는 부분의 표현 등과 같이 복잡한 DTD 표현이 어렵다.

SGML의 DTD 설계를 위한 도구의 대표적인 예로, 광운대에서 제시한 SGML Editor가 있다[10]. 이 편집기는 기본 과서의 구현에 중점을 둔 것으로서 텍스트로 설계된 DTD를 태그 편집기를 통해 수정하는 방식을 취하고 있다. SoftQuad사에서 만든 SGML 편집기인 Author/Editor[11]는 문맥 의존적으로 설계된 틀로서 텍스트로 설계된 DTD를 RulesBuilder라는 프로그램으로 미리 컴파일한 DTD를 이용한다. 그러므로 문법에 어긋난 태그의 삽입이 불가능하다[12]. 따라서 앞의 두 도구들은 태그 위주의 시각적인 기능만을 제공함으로써 DTD의 전체적인 계층 파악이 미흡하며 일반 사용자의 DTD 설계가 어렵다.

Microstar의 Near & Far Author는 DTD 구조의 그래픽 매핑 기법을 사용한 도구로서 사용자가 직접 DTD를 설계하지 않고 기존의 것을 반입하거나 DTD 설계를 위한 디폴트 맵을 사용하여 설계한다. SGML DTD 표현 방식은 아이콘 기반의 인터페이스를 지원하는 대화식 구조 트리 형식이며 구조 트리의 기호는 아이콘에 따른 사건과 값의 내용을 보여줌으로써 엘리먼트의 출현 빈도수의 파악이 용이하며 전체적으로 확장할 경우 분류 체계, 사건, 디폴트 값 내용을 파악할 수 있다. 그러나 구조 트리 설계 기법을 사용함으로써 XOMT가 제공하는 DTD 전반에 걸친 구성 요소에 대한 시각화와 객체 다이아그래밍 기법적인 요소는 없다.

HyTime의 DTD 설계를 위한 대표적인 도구로는 TechnoTeacher사의 제품을 들 수 있다[13]. 이것은 HyTime 엔진이 탑재된 HyMinder, SGML 저장소인 MarkMinder, SGML/HyTime 문서를 위한 브라우저와 에디터를 갖춘 HyBrowse HyTime Browser로 구성되어 있다. 아직 초기의 도구로서 HyTime DTD 설계는 순수 텍스트 에디터에 의존하기 때문에 일반 사용자의 사용은 매우 어렵다.

본 논문에서는 HyTime DTD를 설계할 수 있는 객체 다이아그래밍 기법인 HOMT를 제안한다. 이 HOMT는 SGML DTD를 설계하는 객체 다이아그래밍 기법인 XOMT[3]를 확장하였으며 SGML 뿐만 아니라

Hy Time DTD도 보다 쉽게 설계할 수 있고 관리 또한 용이하여 버전문가도 쉽게 DTD를 설계할 수 있다는 장점이 있다. HOMT는 HyTime을 지원하기 위하여 XOMT에 선언부를 추가하였으며 하이퍼미디어 응용들의 특성인 다양한 미디어들 사이의 관련성과 시간성을 지원하는 HyTime 여섯 개 모듈의 아키텍처 폼(Architectural form)을 상속할 수 있도록 확장하였다.

3. SGML과 HyTime

3.1 SGML

SGML은 이기종 시스템들 간의 전자 문서 교환을 위해 마크업 언어 구문을 정의하는 메타 언어(meta language)이다[6]. 이 SGML을 이용하여 문서를 작성하기 위해서는, 우선 문서를 분석해서 구조를 인식한 후, 마크업을 이용하여 정의하고, 이 마크업에 맞춰서 실제 문서를 기술하는 과정이 필요하다.

SGML 문서는 크게 SGML 선언부(Declaration), 문서형 정의부(Document Type Definition, DTD), SGML 문서 실례부(SGML Document Instance)로 이루어진다.

SGML 선언부는 문서가 한 시스템에서 다른 시스템으로 전송되었을 경우 시스템간에 사용하는 문자 코드가 같도록 하고, 특별한 역할을 하는 코드에 대해 해석이 같도록 하기 위해서 사용된다. 이를 위해 선언부에서는 SGML 문서에서 이용하는 문자 집합, 문서에서 특수한 기능을 하는 문자에 대한 설명, 그리고 문서에서 사용되는 여러 기능 등을 기술한다.

SGML의 문서 형 정의부는 프로그래밍 언어를 정의하기 위해 BNF 표현을 사용하는 것처럼 하나의 일반적 마크업 언어를 정의한다. 이 정의는 문서의 전체적인 논리적 구조를 결정한다. 마크업 언어의 정의 방법은 컴파일러의 문법 정의와 유사한 형태이며, SGML 선언부에서 정의된 형식에 맞춰 작성해야만 한다.

SGML 문서 실례부는 정의된 문서 형 정의부에 맞추어 실제적인 문서의 내용을 나타내는 부분과 추가적으로 마크업 정보를 삽입하는 부분이다. 사용자가 입력을 쉽게 하기 위해서 엔티티와 생략 태그가 이용된다. 생략 태그에 대해서는 SGML 파서(parser)가 파싱(parsing) 과정 중에 전후 엘리먼트를 분석하여 복구하게 된다.

전자 문서를 작성하기 위해서는 문서의 논리적 구조를 나타내는 DTD를 우선 기술하여야 한다. DTD의 주요 구성 요소에는 엘리먼트(ELEMENT), 엔티티(ENTITY), 속성리스트(ATTLIST)가 있으며, 그외 각각의 엘리먼트나 엔티티가 어떻게 처리되는지를 나타내는 처리 명령어(processing instruction) 선언, 그래픽 데이타 등을 처리하기 위한 노테이션(notation) 선언, 마크업을 보다 쉽게 하기 위한 단축 참조표(short reference map) 선언 등이 있다.

3.2 HyTime

HyTime[5, 7, 8]은 하이퍼미디어 문서의 효율적인 관리를 위한 SGML의 응용, 확장, 완성이라는 의미로 정의된 하나의 응용 구조이며, SGML의 구문을 사용하여 기술된 메타 DTD로서 HyTime에 적합한 모든 응용은 SGML에 적합한 응용이 된다. 또한, 아키텍처 폼(architectural forms)이라고 불리어지는 SGML 엘리먼트 타입과 속성리스트를 위한 틀의 집합(set of templates)으로 정의된 메타 문서 형식(meta document type)이다. HyTime 메타 DTD의 규칙 집합은 아키텍처 폼에 의해 표현된다.

HyTime은 다양한 응용을 위한 요구 충족을 위해 12개의 구(clauses)와 3개의 부록(annexes)을 정의함으로써 체계적인 모듈 분리를 하고 있으며, 아키텍처 폼은 6개의 모듈(six modules)로 구성되어 있다. 각 모듈은 기본적인 기능 집합과 하나 이상의 선택적인 특징을 포함한 아키텍처 폼이다.

여섯개의 모듈은 Base 모듈, Measurement 모듈, Location Address 모듈, Hyperlinks 모듈, Scheduling 모듈, Rendition 모듈로 이루어져 있다. Base 모듈은 기본 모듈로서 HyTime 문서를 위한 최소의 기능을 제공하며, 많은 선택적인 기능을 제공한다. Measurement 모듈은 위치(position)와 객체의 크기(extent of objects)를 정의하는 기능을 제공한다. 즉, 좌표 공간에서의 주소 지정 방식과 좌표 주소의 구성 방식을 정의하고 있다. Location Address 모듈은 위치의 주소 지정 방식을 정의하고 있으며, 위치 주소는 하이퍼링크의 생성에 사용된다. Hyperlinks 모듈은 정보 엘리먼트와 독립된 문서 계층들 사이의 관계를 기술하며, ilink (independent link), clink(contextual link), 그리고 nlink(notation specific link) 등 하이퍼링킹 아키텍처 폼을 정의하고 있다. Scheduling 모듈은 유한 좌표 공간과 이들 공간에서 발생하는 사건 스케줄을 위한 아키텍처 폼을 정의하고 있다. Rendition 모듈은 Sche

```
        FORMAL   YES
        APPINFO  ArcBase

        <!AFDR "ISO/IEC 10744:1992" >
        <?ArcBase HyTime >
        <!Notation HyTime PUBLIC "ISO/IEC 10744/1992//NOTATION HYTIME ARCBASE
                                   HyTime Architecture Definition Document//EN" >
        <!Attlist #NOTATION HyTime
                ArcFormA     NAME  HyTime
                ArcNamrA     NAME  HyNames
                ArcSuprA     NAME  HyBrid
                ArcDocF      NAME  HyDoc
                ArcDTD       CDATA "HyTime"
                ArcBridF     NAME  HyBrid
                ArcAuto      (ArcAuto|nArcAuto) ArcAuto
                ArcIndr(     ArcIndr|nArcIndr) nArcIndr
                ArcOptSA     NAMES "base locs links"
                base         CDATA "base autobos exidrefs"
                locs         CDATA "locs anydtd anysgml mixcase multiloc bibloc nameloc"
                links        CDATA "links"   >
        <!NOTATION AFDRMeta PUBLIC "ISO/IEC 10744//NOTATION AFDR Meta DTD Notation//EN"   >
        <!Entity HyTime PUBLIC "ISO/IEC 10744//DTD AFDR Meta-DTD Notation//EN"
                              CDATA AFDRMeta   >
```

(그림 1) BibCat 메타 DTD[7]
(Fig. 1) BibCat Meta DTD[7]

duling 모듈이 사용될 시 사건 해석 기술을 위한 아키텍처 폼을 정의하고 있다.

HyTime 문서는 특정 응용을 위해 필요한 모듈들의 집합으로 구성되며, 각 모듈은 HyTime 아키텍처 폼의 집합을 지원한다. 이들 중 Base 모듈은 항상 요구되며, 다른 모듈들은 선택적이거나, 특정 모듈에 의존적인 성향을 가지고 있다.

3.3 HyTime의 지원을 위한 SGML의 확장

HyTime을 지원하기 위한 SGML의 확장은 문서내에 유도되어질 HyTime의 아키텍처들을 연결하는 메카니즘을 이용한다. 이 메카니즘은 노테이션이나 엔티티를 이용하여 문서와 아키텍처를 연결하고 문서에 사용되는 모듈의 집합이나 선택적인 아키텍처를 선언하며, 처리할 메타 DTD와 연결한다[7].

보다 구체적으로 기술하면, SGML 선언부의 **FORMAL** 파라메타를 **YES**로 대치하는 것은 외부 객체와의 연결을 의미하며, **APPINFO**(application specific information) 파라메타에 **"APPINFO ArcBase"** 혹은 **"APPINFO "HyTime"**을 선언하는 것은 HyTime 아키텍처를 문서에 사용하는 처리 시스템(processing system)을 의미한다. 또한 처리 명령부에 HyTime 모듈에 관련된 사항을 기술함으로써 문서내에 사용될 모듈을 정의한다.

그림 1은 노테이션과 엔티티를 이용하여 HyTime 아키텍처와 연결한 예로써 노테이션의 속성부에 DTD 내에 사용될 명세를 기술한다.

<!AFDR "ISO/IEC 10744:1992" > 이라고 선언하면 AFDR(Architectural Form Definition Requirements)의 사용을 기술한 것으로서, 이것은 HyTime을 이용한 새로운 메타 DTD를 생성한다는 의미이다.

```
FORMAL   YES
APPINFO  "HyTime"

<?HyTime VERSION "ISO/IEC 10744:1992" HTQCNT=32 >
<?HyTime MODULE base context >
<?HyTime MODULE locs pathloc multloc coordloc >
<?HyTime MODULE measure >
<?HyTime MODULE links >

<!DOCTYPE techdoc [
<!ENTITY % loc   "nameloc|pathloc" >
<!ENTITY % link "clink|ilink" >
          .........
        ]>
```

(그림 2) HyTime 문서(techdoc)
(Fig. 2) HyTime Document(techdoc)

그림 2는 techdoc이라는 HyTime 응용 DTD의 일부분을 기술한 것으로서 Base, Location Address, Measurement, Hyperlinks 모듈을 연결한 예이다.

4. HyTime의 지원을 위한 XOMT의 확장(HOMT)

이 장에서는 SGML DTD 설계를 지원하는 XOMT의 기본 기능과 이의 미비점을 보완하는 확장 기능을 기술하며, 특히 HyTime을 지원하기 위해 고안된 새로운 다이아그래밍 기법인 HOMT(HyTime support XOMT)를 다룬다.

4.1 XOMT의 기본 기능

이 절에서는 SGML의 DTD를 기술하기 위해 OMT [9]의 객체도를 확장시킨 XOMT 다이아그램 표기법을 예와 함께 개괄적으로 기술한다. 보다 상세한 설명은 XOMT에 대한 논문을 참조하기 바란다.

먼저 XOMT의 SGML DTD 표기법을 살펴보기 위해 그림 3은 일반 메모지 형식에 대한 간단한 SGML DTD를 기술한 것이다.

그림 4는 그림 3의 DTD를 XOMT도로 표현한 것으로서, 크게 세 종류의 부윈도우를 이용하여 DTD를 설계한다. 이들은 DTD의 주 트리 부분이 그려질 Main 부윈도우, 각각의 Entity 정의시 마다 하나씩 발생될 Entity 부윈도우, 원쪽 참조가 발생될 때 내부적으로 인덱스를 관리하면서 발생될 TMP 부윈도우이다.

전체적으로 엔티티 정의를 XOMT 다이아그램으로 표현한 다음, 실제 문서의 논리적인 구조를 이루는 엘리먼트에 대한 구조를 기술한다. 끝으로 해당 엘리먼트의 속성들을 처리함으로서 DTD를 완성한다.

4.2 XOMT의 확장 기능

SGML DTD내에는 수학적인 기호나 그래픽 데이타와 같이 다른 언어로 기술된 데이타를 처리할 수 있도록 하는 노테이션(NOTATION)과 특정 엘리먼트의 내용 모델속에 그 자신의 엘리먼트가 포함되어 있는 자기 참조(self reference)가 자주 등장한다. 그러나 앞에서 살펴본 XOMT의 기본 기능에는 이에 대한 표기법이 없다. 이는 특히 HyTime을 지원하기 위해서 반드시 있어야 한다. 본 절에서는 이들에 대한 표기법을 새로이 제시한다.

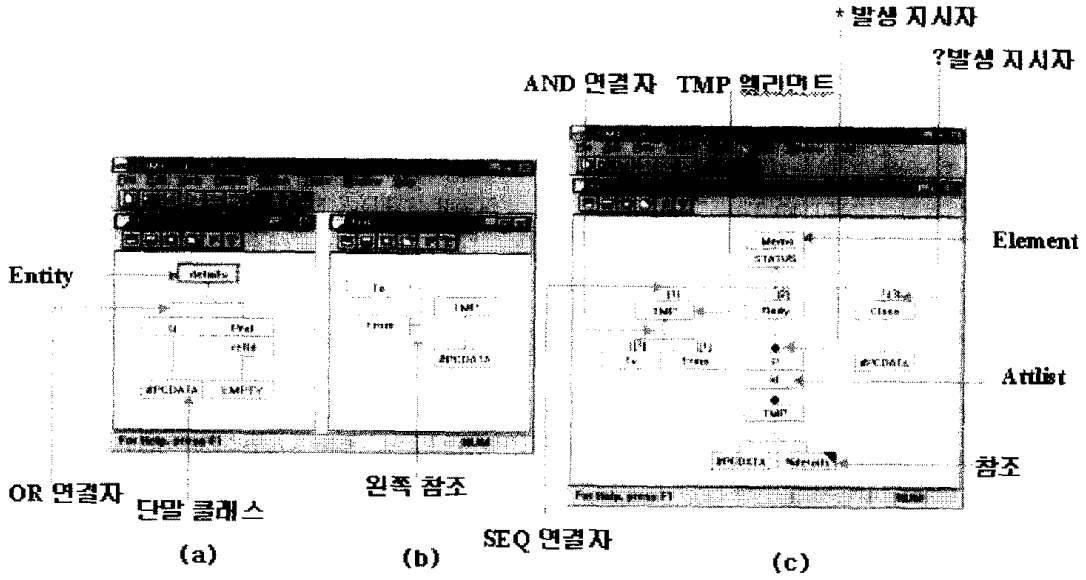
4.2.1 노테이션과 노테이션의 속성 지정

노테이션은 SGML이 아닌 Tex, 캡슐화된 PostScript(Encapsulated PostScript) 등에 의해 정의된 마크업 규칙을 따르는 문서의 특성 부분을 가리키기 위한 방식으로 수학적 기호나 그래픽 데이타의 표기에 사용되는 DTD의 구성 요소이다[6].

이러한 노테이션은 계층 구조를 가지지 않고 참조만 하면 되기 때문에 노테이션 부윈도우를 추가하여

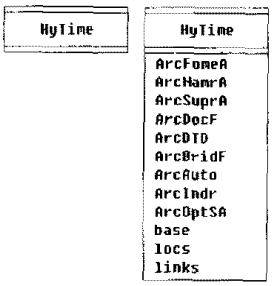
```
<!-- DTD for simple office memoranda                                    -->

<!ENTITY % details  "Q|Pref"                                    >

<!--      ELEMENTS      MIN      CONTENT (EXCEPTIONS)                  -->
<!ELEMENT  Memo          -        ((To & From), Body, Close?)           >
<!ELEMENT  (To|From)     - O      (#PCDATA)                             >
<!ELEMENT  Body  - O     (P*)                                  >
<!ELEMENT  P     - O     (#PCDATA|%details;)*                  >
<!ELEMENT  Q     - -     (#PCDATA)                             >
<!ELEMENT  Pref  - O     EMPTY                                 >
<!ELEMENT  Close - O     (#PCDATA)                             >

<!--      ELEMENTS      NAME     VALUE          DEFAULT               -->
<!ATTLIST  Memo STATUS   (confiden|public)      public                >
<!ATTLIST  P             id       ID            #IMPLIED              >
<!ATTLIST  Pref  refid   IDREF    #REQUIRED                  >
```

(그림 3) 메모를 위한 DTD
(Fig. 3) DTD for simple office memoranda



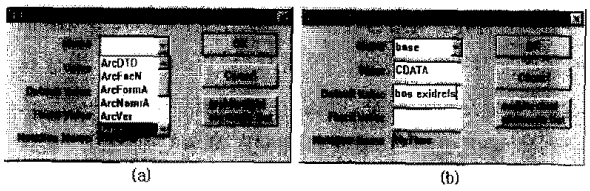
(그림 4) MEMO DTD의 XOMT도 (a) Entity 부윈도우 (b) TMP 부윈도우 (c) Main 부윈도우
(Fig. 4) XOMT diagram for MEMO DTD (a) Entity subwindows (b) TMP subwindows (c) Main subwindows

선부 독립적으로 선언한다. 노테이션의 표기는 상하 이중선을 가진 사각형으로 표현하며, 속성은 엘리먼트의 속성 표현 방식과 유사하다. 그림 5는 HyTime 문서 표현을 위한 노테이션이다.



(그림 5) HyTime 아키텍쳐 노테이션
(Fig. 5) HyTime Architectual Notation

노테이션의 속성 지정은 엘리먼트에 대한 속성 지정과 같은 방식으로 정의한다. 단, 속성리스트 다이알로그 박스는 구분된다. 그림 6(a)는 HyTime의 아키텍처 노테이션의 표현인데, 이는 정해진 속성에 대한 명세만을 요구함으로 리스트박스에 이미 기억되어 있는 해당 이름을 선택하거나 직접 입력한다. 일반적인 노테이션에 속성을 정의할 경우는 Name(현재 콤보박스) 부분에 직접 속성명을 입력한다. 그림 6(b)는 아키텍처 노테이션의 Base 모듈을 선택한 경우이다.



(그림 6) 노테이션의 속성 지정
(Fig. 6) Designation of Notation's attributes

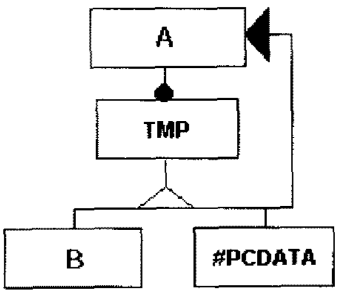
한편 엔티티에서 노테이션을 이용할 경우는, 먼저 엔티티를 이전 방식과 같이 선언한 후에 객체 툴바에서 노테이션을 선택한 후 해당 엔티티에 클릭하면 서로의 관계를 정의하는 것으로 한다.

4.2.2 자기 참조 지정

DTD의 엘리먼트는 문서의 구조에 대한 논리적인 요소로서, 엘리먼트의 내용 부분은 내용 모델(content model)과 이미 선언된 내용(declared content)으로 기술된다. 내용 모델은 엘리먼트 이름인 내용 토큰 리스트들이 연결자나 발생 지시자들과 함께 나타나는데, 이때 내용 토큰 리스트에는 엘리먼트와 같은 토큰 리스트가 발생할 수 있다. 이를 자기 참조라 한다. 또한 엔티티의 경우도 같은 이름을 사용하는 경우가 발생한다.

이러한 자기 참조 지정을 위해 연결자 끝에 검은색 삼각형을 결합시켜 표현하였다.

그림 7은 **<!ELEMENT A - - (B|#PCDATA|A)* >**을 표현한 것으로 하위 엘리먼트(혹은 엔티티 참조)들이 '|'(OR)의 관계로 상위 엘리먼트(혹은 엔티티 참조)를 구성함을 의미하며 하위 엘리먼트와 상위 엘리먼트간에 is-a 관계성(삼각형)으로 나타난다. 그런데 두 개 이상의 하위 엘리먼트들이 묶여져서 발생 지시자(* : 0 또는 이상)와 함께 나타났기 때문에 임시 엘리먼트 클래스(키워드로 TMP 클래스)를 두어 관계성을 모델링한 것이다. 하위 엘리먼트에 상위 엘리먼트와 같은 엘리먼트가 존재하므로 자기 자신 참조 기법이 필요하다. 그러므로 새롭게 제안된 연결자가 상위 엘리먼트를 가리킨다.



(그림 7) 자기 참조
(Fig. 7) Self-reference

4.3 HOMT 도구

HOMT(HyTime Support XOMT) 도구는 XOMT가 HyTime을 지원할 수 있도록 확장한 아이콘 기반의 하이퍼미디어 문서 설계 지원 도구이다. 이는 기본적으로 SGML DTD를 지원하는 XOMT의 기능을 모두 사용할 뿐만 아니라 HyTime의 6개 모듈들을 용이하게 HyTime DTD 설계에 사용할 수 있도록 한다.

HOMT 도구의 주요 기능들을 살펴보기 위해 전형적인 HyTime DTD를 예로 들어 본다. 그림 8은 참고문헌 관리를 위한 HyTime DTD이다. 이는 메타 DTD로서 참고 문헌 관리를 위해 새로운 응용 DTD 설계시 이 구조를 상속받아 구현할 수 있다.

4.3.1 선언부

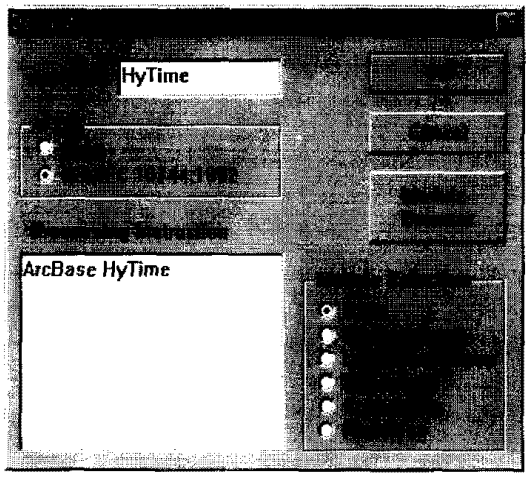
HyTime DTD를 설계할 때 선언부의 기술은 필수적이며, 이를 위해 HOMT 도구에서는 그림 9와 같은 다이알로그 박스를 이용하여 선언부에 기술될 항목을 지정하여 입력하도록 한다. 그림 9는 그림 8의 ***(A)*** 부분을 표현한 것이다.

APPINFO 부분은 사용자가 직접 입력하며, AFDR (Architectural Form Definition Requirements)의 사용 여부 선택과 DTD에서 사용될 모듈들에 대한 선택이 가능하다. 또한 처리 명령(Processing Instruction)은 선언할 만큼 줄을 구분해서 계속 입력한다.

```
        <!   Meta DTD for the BibCat architecture      -->
(A)     <!AFDR "ISO/IEC 10744:1992" >
        <?ArcBase HyTime >
(B)     <!Notation HyTime PUBLIC "ISO/IEC 10744/1992//NOTATION HYTIME ARCBASE
                                  HyTime Architecture Definition Document//EN"   >
        <!Attlist #NOTATION HyTime
                ArcFormA    NAME  HyTime
                ArcNamrA    NAME  HyNames
                ArcSuprA    NAME  HyBrid
                ArcDocF     NAME  HyDoc
                ArcDTD      CDATA "HyTime"
                ArcBridF    NAME  HyBrid
                ArcAuto     (ArcAutolnArcAuto) ArcAuto
                ArcIndr     (ArcIndr|nArcIndr) nArcIndr
                ArcOptSA    NAMES "base locs links"
                base        CDATA "base autobos exidrefs"
                locs        CDATA "locs anydtd anysgml mixcase multiloc bibloc nameloc"
                links       CDATA "links"    >
        <!NOTATION AFDRMeta PUBLIC "ISO/IEC 10744//NOTATION AFDR Meta DTD
                Notation//EN"   >
        <!Entity HyTime PUBLIC "ISO/IEC 10744//DTD AFDR Meta DTD Notation//EN"
                                  CDATA AFDRMeta   >
(C)     <!ATTLIST #ALL            ID     ID   #IMPLIED
                natlang NAME #IMPLIED   >
        <!ENTITY % fields "Title|Author|Publisher"   >
        <!ENTITY   %leaf content "#PCDATA|BibBrid|See Also" >
        <!ELEMENT  BibDoc      - O (GenTitle?, BibEntry*)      -(Nameloc)    >
```

```
(D)     <!ATTLIST  BibDoc HyTime NAME #FIXED "HyDoc" >
        <!ELEMENT  BibBrid        (%leaf content;)* >
        <!ELEMENT  GenTitle       O (%leaf-content;)* >
        <!ELEMENT  BibEntry       O (%fields;)* >
        <!ATTLIST  BibEntry pubtype (book|serial|paper|bibpla|article|undefined) undefined >
        <!ELEMENT  (%fields;)     - (%leaf content;)* >
        <!ELEMENT  See Also       O (%leaf-content;)*       (See Also) >
        <!ATTLIST  See Also HyTime   NAME  #FIXED "clink"
                                  anchaddr IDREF #REQUIRED >
        <!ELEMENT  Nameloc        (#PCDATA) >
(E)     <!ATTLIST  Nameloc ID        ID                  #REQUIRED
                                  locsrc    ENTITY            #IMPLIED
                                  deflocsc (elements|entities) elements
                                  HyTime    NAME              #FIXED "nameloc" >
        <! - End of BibCat meta-DTD -->
```

(그림 8) 참고 문헌 관리를 위한 메타 DTD[7]
(Fig. 8) Meta-DTD for the BibCat architecture[7]

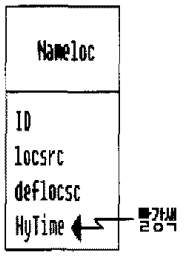


(그림 9) 선언부 다이알로그 박스
(Fig. 9) Declaration dialogue box

메타 DTD의 설계와 응용 DTD의 설계는 **AFDR**의 선택에 의해 구분되며, HyTime의 Base 아키텍처의 사용과 이미 만들어진 메타 DTD의 사용 여부는 처리 명령(**Processing Instruction**)의 ArcBase에 포함된 내용에 의해 구분된다. 이 내용은 계속 유지되며, 속성 리스트의 속성명의 선택과 아키텍처 엘리먼트 리스트(**architectural elements list**) 다이알로그 박스의 항목으로 이용된다.

4.3.2 엘리먼트의 HyTime 속성 표현

엘리먼트가 HyTime 속성을 가지면 이것은 빨강색으로 표시된다. 그림 10은 그림 8의 ***(E)*** 부분을 표현한 것이다.

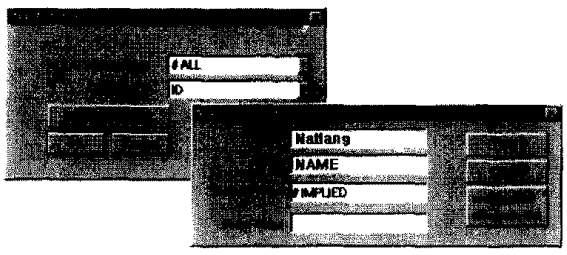


(그림 10) HyTime 속성 표현
(Fig. 10) Representation of HyTime attribute

4.3.3 #ALL의 표현

#ALL과 같이 DTD내의 모든 엘리먼트들에 대한 일괄적인 속성 지정이나 특정 부류 엘리먼트들만을 위한 속성 지정을 위해 그림 11과 같은 다이알로그 박스를 이용하여 입력하도록 한다. 먼저 엘리먼트를 위한 이름을 입력한 후, 추가 속성(ADD ATTRIBUTE) 버튼을 눌러서 엘리먼트의 속성 지정 방식과 같이 계속 입력하면 된다.

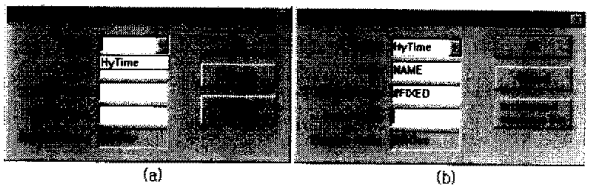
그림 11은 그림 8의 ***(C)*** 부분을 표현한 것이다.



(그림 11) #ALL의 속성 지정
(Fig. 11) Designation of #ALL's attributes

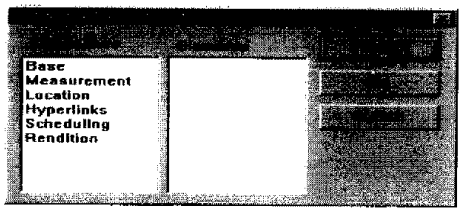
4.3.4 HyTime 지원 속성 지정

그림 8의 ***(D)***와 ***(E)*** 부분처럼 HyTime의 DTD를 설계할 때 몇몇 엘리먼트들은 6개의 아키텍쳐 모듈이 가지고 있는 속성을 상속받아 사용하게 된다. 이를 지원하기 위해 HOMT 도구에서는 XOMT에서의 속성 표현 방식과는 달리 Name부에 HyTime이 고정되어 있게되며(그림 12(a) 참조), HyTime 선택시 값(**Value**)은 "**NAME**", 디폴트 값(**Default Value**)은 "**#FIXED**"로 자동 지정된다(그림 12(b) 참조).



(그림 12) HyTime 지원 속성 지정
(Fig. 12) Designation of HyTime supporting attributes

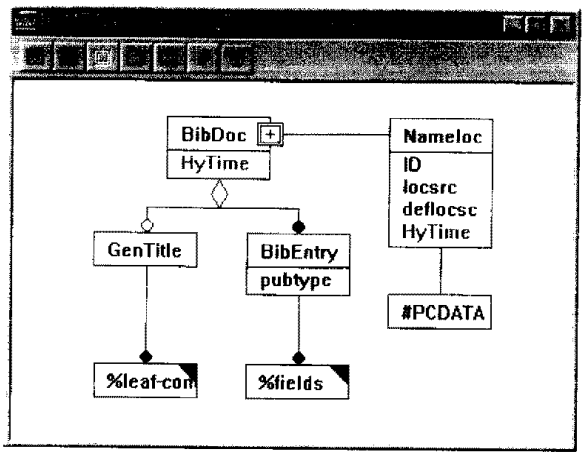
선언부에서 정의한 각 모듈들에 대한 정보를 기초로 한 고정값 지정시 선택된 아키텍쳐 엘리먼트 리스트를 통해 HyTime 모듈의 엘리먼트 정보가 HyTime 스키마로부터 적재되며 사용자가 지정한 엘리먼트의 기본적인 속성은 HyTime 모듈로부터 상속된다. 또한, 사용자는 상속받은 속성을 DTD에 맞게 재편성할 수 있다.



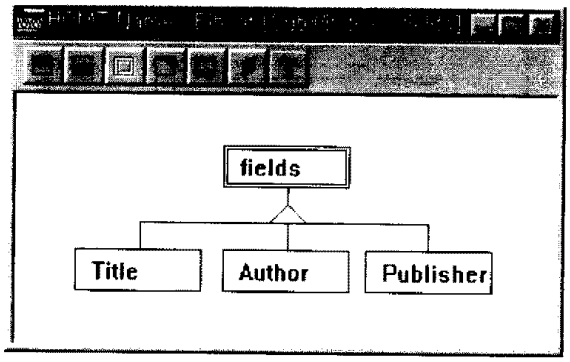
(그림 13) 아키텍쳐 엘리먼트 리스트의 사용
(Fig. 13) Use of Architectual elements list

5. 적용 예

본 장에서는 4장의 HyTime DTD(그림 8)에 대해 HOMT 도구를 적용한 예를 알아본다. HOMT 도구를 이용한 HyTime DTD의 설계는 크게 4종류의 부윈도우를 이용한다. 이 중 Main 부윈도우(그림 14)는 DTD의 주 트리부를 설계하는 부윈도우로서 최상위 엘리먼트로부터 설계된다. 이때 엔티티 참조와 TMP 참조가 이루어진다.



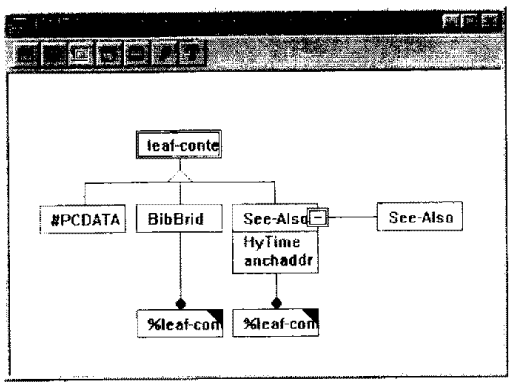
(그림 14) Main 부윈도우
(Fig. 14) Main subwindows



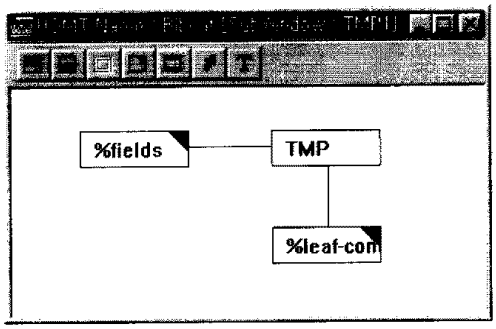
(그림 15) fields Entity 부윈도우
(Fig. 15) fields Entity subwindows

그림 8에 나타나는 두 개의 엔티티는 각각의 엔티티 이름을 가지는 부윈도우에서 설계한다. 그림 15는 엔티티 fields를, 그림 16은 엔티티 leaf-content를 표현한 것이다. leaf content 엔티티는 #PCDATA, BibBrid, 그리고 See-Also를 하위 클래스(is-a 관계성)로 가지며 엘리먼트 BibBrid와 See-Also는 엔티티 leaf-content를 다시 참조한다. 엘리먼트 See Also는 Hyperlinks 모듈의 clink를 상속받으며 See-Also를 EXCLUDE한다.

TMP 부윈도우는 엘리먼트 이름부가 엔티티 참조이거나 두 개 이상의 엘리먼트로 연결되어 나타날 경우 이를 표현한다. 그림 17은 그림 8에 나타나는 **<!ELEMENT (%fields;) - (%leaf-content;)* >**를 표현한 것으로 엘리먼트 이름부에 엔티티 참조가 발생하기 때문에 왼쪽 참조 표현 방식을 사용한다. 이때 TMP 부윈도우의 인덱스는 자동으로 부여된다.

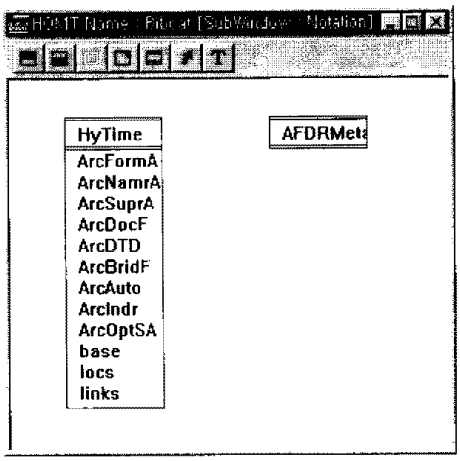


(그림 16) leaf-content Entity 부윈도우
(Fig. 16) leaf-content Entity subwindows



(그림 17) TMP1 부윈도우
(Fig. 17) TMP1 subwindows

Notation 부윈도우는 DTD내의 노테이션과 이의 속성을 표현하는 윈도우로서 그림 18은 그림 8의 ***(B)***부분을 표현한 것이다. 노테이션의 속성은 그림 6과 같이 지정되며 그 정보는 내부 화일에서 관리된다. 윈도우상에는 노테이션명과 속성명만이 나타난다.



(그림 18) Notation 부윈도우
(Fig. 18) Notation subwindows

6. 결론 및 향후 과제

본 논문은 HyTime DTD를 다이아그램 형태로 설계할 수 있도록 XOMT도를 확장한 HOMT도를 제안하였다.

하이퍼미디어 응용 시스템의 개발시 객체 다이아그래밍 기법인 HOMT를 이용하여 SGML과 HyTime의 DTD를 쉽게 설계할 수 있으며, 설계된 DTD는 객체지향 데이타베이스에서 통합적으로 관리된다. HOMT는 현재 제기되고 있는 하이퍼미디어 문서 처리의 문제점인 가변적인 텍스트 엘리먼트의 다양한 크기, 서브 엘리먼트들의 공유, 분류 체계에 따른 관계 횡단에 대한 잦은 요구 등 하이퍼미디어 응용 분야에서 대두되는 여러 어려움을 설계 단계부터 극복하는 해결책이 될 것이다.

현재 XOMT를 확장한 HOMT의 설계를 완료하고 개발이 진행중이다. 앞으로 객체지향 데이타베이스에서 HyTime 문서를 관리할 수 있도록 XOMT 데이타베이스 스키마도 확장하여야 한다.

최근 SGML과 HyTime을 기반으로 많은 종류의 언어들이 표준화되고 있다. 그 대표적인 것으로 DSSSL (Doc-ument Style Semantics and Specification Language, ISO/IEC 10179), SMSL(Standard Hypermedia/Multimedia Scripting Language, ISO/IEC 13240), SMDL(Standard Music Description Language, IS-O/IEC 10743) 등이 있으며, 앞으로 이런 언어들로 기술된 DTD들도 완벽하게 표현할 수 있는 객체 다이아그래밍 기법이 개발되어야 한다.

참 고 문 헌

[1] W. Kim, "Object-Oriented Database : Definition and Research Directions," IEEE Transactions on Knowledge and Data Engineering, Vol.3, No.1, pp.327-341, Sept. 1990.

[2] J. Zhang, "Application of OODB and SGML Techniques in Text Database : An Electronic Dictionary System," SIGMOD RECORD, Vol.24, No. 1, pp.3-8, March 1995.

[3] 박인호, 한에노, 정은주, 김은정, 배종민, 강현석, 김완석, "XOMT : SGML DTD 설계를 위한 객체 다

이아그래밍 기법," 한국정보과학회 논문지, 제3권, 제3호, pp.228-237, 1997. 6.

[4] 한예노, 박인호, 강현석, 김완석, "SGML 문서의 관리를 위한 객체지향 데이터베이스 설계," 한국정보처리학회 논문지, 제4권, 제3호, pp.670-684, 1995. 3.

[5] ISO, *Hypermedia/Time-based Structuring Language : HyTime(ISO 10744)*, 1992.

[6] ISO, *International Standard ISO/IEC8879 : Information Processing-Standard Generalized Markup Language(SGML)*, Geneva/New York, 1986.

[7] W. Kimber, *Practical Hypermedia : An Introduction to HyTime*, 1996.

[8] C. Goldfarb, *Catalog of HyTime Architectural Forms and HyTime SGML Specification Version 2.0*, June 1993.

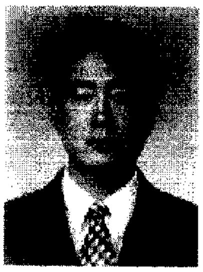
[9] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

[10] 현득창, 임광택, 이수연, "SGML 기반 파서를 이용한 SGML 문서 편집기의 구현," 한국정보과학회 논문지, 제20권, 제4호, pp.484-494, 1993. 4.

[11] SoftQuad Author/Editor 3.5, [URL:http://www.sq.com/products/authored/]

[12] 한반신, 고영곤, 최윤철, "문법 기반 범용 SGML DI 편집기의 설계 및 구현," 한국정보과학회 논문지, 제3권, 제3호, pp.321-331, 1997. 6.

[13] TechnoTecher, Inc., [URL: http://www.techno.com/]

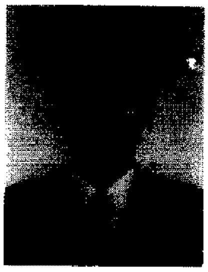


장 원 호

1996년 경상대학교 컴퓨터과학과 (이학사)
1998년 경상대학교 전자계산학과 (공학석사)
1998년~현재 현대상선 재직중
관심분야 : 객체지향 데이터베이스, SGML/HyTime, 전자문서관

임 혜 정

1994년 창원대학교 전자계산학과 (이학사)
1996년~현재 경상대학교 전자계산학과 석사과정 재학중
관심분야 : 객체지향 데이터베이스, SGML/HyTime



박 인 호

1990년 경상대학교 전산통계학과 (이학사)
1997년 경상대학교 전자계산학과 (공학석사)
1997년~현재 경상대학교 전자계산학과 박사과정
관심분야 : 객체지향 데이터베이스, SGML/HyTime/XML, 멀티미디어

강 현 석

1981년 동국대학교 전자계산학과 (학사)
1983년 서울대학교 계산통계학과 (이학석사)
1989년 서울대학교 계산통계학과 (이학박사)
1981년~1984년 한국전자통신연구원 연구원
1984년~1993년 전북대학교 전자계산학과 부교수
1993년~현재 경상대학교 컴퓨터과학과 교수
관심분야 : 객체지향 데이터베이스, 컴퓨터 그래픽스, 멀티미디어