SGML에 기반한 하이퍼미디어를 위한 HyTime 엔진 설계

o
권일정, 신경희, 유재우
숭실대학교 컴퓨터학부

HyTime Engine Design for Hypermedia Systems based on SGML

Il-Jung Kwon, Kyoung-Hee Shin, Chae-Woo Yoo
School of Computing, Soongsil University

요 약

HyTime(Hypermedia/Time-Based Structuring Language)는 SGML문서에 하이퍼미디어를 포함하는 기능으로 확장하고, 하이퍼미디어의 논리구조를 기술하기 위한 언어로 1992년 4월에 국제표준으로 채택되었다. 기존의 문서들은 각기 다른 플랫폼마다 그들만의 문서포맷이 정해져 있는 이유로 상이한 플랫폼에서도 문서를 서로 교환할 수 있도록 하기 위한 SGML 표준을 채택하게 되었다. SGML 환경하에서 많은 문서들은 하이퍼링크와 멀티미디어 정보를 포함하는 문서로 발전하게 된다. 하이퍼미디어문서의 처리를 위한 표준으로 HyTime이 정의되고 많은 어플리케이션들이 이를 적용하고 있으며 하이퍼문서로의 확장을 위한 HyTime 엔진이 요구된다. 이에 본 논문에서는 하이퍼미디어 문서 시스템을 구현하기 위한 필수 요소인 HyTime 엔진의 설계와 HyTime 엔진의 처리절차에 관해 연구하였다.

1. 서론

글을 쓰는 작업을 시작하면서 많은 사람들은 문서를 구별짓기 위해 그들의 문서에 문서명과 저자를 쓰기 시작했다. 그리고 문서와 문서의 구성요소들을 참조할 수 있는 방법들에 대해서 연구하게 되었다. 참조할 수 있는 의미단위로 페이지를 사용하기도 하고 보다 구조적 의미를 지니는 장, 절, 줄 등을 사용하기도 했다. 컴퓨터의 도입으로 이러한 것은 상호 관계있는 문서들을 묶는 방법과 문서내의 특정 객체를 연결하는 방법 그리고 서로다른 플랫폼에서도 문서를 교환할 수 있는 방법을 제안하게 된다. 최근에는 하이퍼미디어나 하이퍼텍스트라는 것을 이용해서 문서가 저장되어 있는 위치에 관계없이 텍스트나 그림, 비디오, 사운드등을 포함하는 문서를 검색할 수 있게 되었다. 이러한 하이퍼미디어 문서를 다루기 위해 1992년 4월에 국제 표준으로 HyTime(Hypermedia / Time-Based Structuring Language)가 제정되었다. HyTime은 문서의 논리 구조를 정의하는 SGML표준의 응용이다. SGML에 있어서 HyTime은 상호 교환과 처리가 가능한 하이퍼문서의 표현을 정의하기 위한 수단을 제공한다. 이에 문서의 논리성을 부여하고 이것을 공통의 문서로 하는 공통된 하위의 시스템을 개발한다면 문서교환을 용이하게 할 수 있을뿐더러 추가비용면에서도 저렴한 방법으로 적용될 것이다. 본 논문은 어플리케이션에 전달되는 문서의 정보를 해석하는 SGML 파서로부터 속성정보를 이어받아 하이퍼미디어 처리를 담당하는 HyTime 엔진의 구성과 설계에 관해 살펴본다.

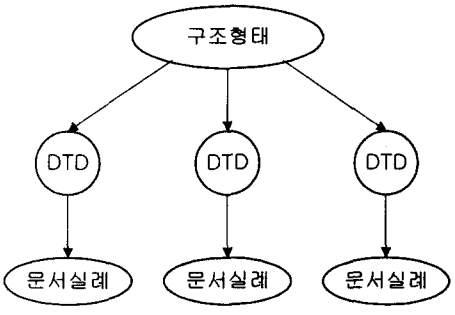
2. SGML

SGML표준은 문서의 구조적 특성을 기술하여 문서의 교환을 용이하게 하였다. SGML문서는 선언부, 문서정의부(DTD), 문서실례(DI)의 3부분으로 구성된다.[2] 문서 선언부는 SGML 문서를 구성하는 문자세트와 구체구문등을 정의하고 문서 전송 시에 참조된다. 문서 정의부는 문서의 논리적인 표현을 마크업을 통해서 이룬다. 그리고 선언을 위한 방법으로 문서유형, 엘리먼트, 엘리먼트의 속성, 엔티티, 표기, 주석등을 이용하게 된다.[2] 문서실례는 문서정의부에서 작성된 DTD (Document type definition)의 마크업 이용해서 실제 텍스트정보를 구성하게 된다.

3. HyTime

HyTime은 어떠한 플랫폼에서도 처리 가능하게 하는 SGML에 하이퍼미디어를 적용할 수 있는 방법을 제시

한다. SGML이 문서의 표현보다는 논리적 구조에 비중을 두는 것처럼 HyTime도 하이퍼미디어문서의 표현보다 문서의 적용에 비중을 두고있다. 이것은 PDF나 postscript등과 같은 특정 표현형태를 갖는 에디터와는 달리 단지 문서의 논리적 구조만을 표현함으로써 가능해진다.[6] HyTime 문서를 사용하기 위해서는 우선 SGML 문서 선언부에 HyTime을 사용하겠다는 것을 명시한다. HyTime은 또한 DTD와 문서 구조를 결정하는 것이 아니기 때문에 어플리케이션 설계자는 자신의 DTD를 만들때에 그것을 기본으로 하는 규칙의 집합을 정해야 한다.[4] 이 규칙을 HyTime에서는 구조형태라고 한다. HyTime은 이것을 통해서 메타DTD를 정의하게 되는 것이다. [그림1]은 HyTime의 구조형태와 DTD 그리고 문서실례의 관계를 나타낸다.

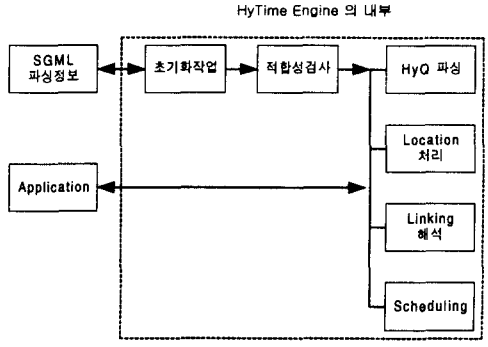


[그림 1] 구조형태와 DTD 및 문서실례와의 관계

HyTime의 구조형태는 기본 모듈, 측정 모듈, 위치지정 모듈, 하이퍼링크 모듈, 스케줄링 모듈, 해석 모듈 6개의 모듈로 구성되어있다. 기본 모듈은 필수로 사용되며 나머지 모듈은 선택적으로 사용될 수 있다. 특히 해석 모듈, 측정 모듈, 스케줄링 모듈 등은 순서대로 의존적이다. 측정 모듈이 사용된다면 반드시 스케줄링 모듈이 사용되어야함을 의미한다. SGML은 문서 내에서 ID와 IDREF를 통해 직접 어드레싱을 가능하게 하지만 특정 문서 내에서만 가능하다. 이에 반해 SGML문서에서 ID속성을 가지는 엘리먼트들에 대해서 HyTime은 구조형태의 ilink를 사용해서 외부문서와의 링크를 설정할 수 있다. 링크의 설정은 앵커라고하는 목적 어트리뷰트명을 지정하는 지정자를 두는 것으로 구현이 가능하다. HyTime은 SGML과는 달리 간접 어드레싱을 가능하게 한다. 엔진에서 사용되는 HyQ 언어는 위치모듈에서 지원하며 원격 앵커를 명시하는 기능을 수행한다. 즉 질의를 통한 간접 어드레싱이라 할 수 있다.

4. HyTime 엔진의 설계

DTD가 HyTime에 적합한가의 검사는 SGML 파싱 과정중에 또는 HyTime 서비스 시에 선택적으로 할 수 있다. 본 논문에서는 SGML파서의 DTD 파싱과정과는 독립적으로 SGML문서 파싱후에 HyTime 서비스와 HyTime 처리 작업등을 포함시키는 것을 전제로 한다. 이것은 HyTime엔진을 SGML 파서와 독립적으로 수행할 수 있도록 한다. 이는 범용 SGML 파서의 확장성을 고려하기 위한 것이기도 하다. SGML 파싱의 단계와 구분을 짓는 또하나의 이유는 구현의 편의성을 들 수 있다. 본 논문에서 설계한 HyTime 처리 시스템에의 HyTime 엔진의 구성은[그림 2]와 같다.



[그림 2] HyTime 엔진의 구성

SGML 파싱정보를 토대로 HyTime 엔진은 해당 파서에 대한 문서가 HyTime 처리요구 문서인지를 판단하는 초기화 작업을 거친 후에 HyTime 표준에 적합한 문서인가를 판단하는 적합성검사가 이루어진다. HyQ와 HyFunc 그리고 HyLex등과 같은 특정 데이터의 내용을 찾는 질의형 언어들 중에서 본 논문은 HyQ만을 처리하는 것으로 한다. DI 문서 내에 HyQ 질의가 검색되면 이를 파싱한다. Location 처리는 이름, 카운팅, 질의를 통해 위치를 지정하는 방법중 이름과 질의를 통한 Location만을 처리하는 것으로 한다. Hyperlink 모듈의 Linking 해석은 clink와 ilink의 구조형태로 정의된 엘리먼트의 ID와 linkend 속성을 처리하는 것으로 이루어진다. scheduling 모듈은 특정 객체의 표현을 위해서 시,공간적 좌표를 계산하고 처리하는 멀티미디어 정보를 다룬다. 이렇게 HyTime 엔진에 의해서 처리된 정보들은 SGML 파싱정보와 더불어 어플리케이션에 제공되는 것이다.

5. HyTime엔진의 문서 처리절차

HyTime엔진에 의해서 처리되는 문서는 SGML 파서에 의해서 처리되는 단계와 HyTime엔진에 의해서 처리되는 단계로 구분된다.

(1) DTD가 SGML 표준을 따르는 적합한 문서인지를

검사한다.

(2) SGML파서가 DI를 파싱한다. 이때 파싱된 결과는 트리로 저장된다.

(3) HyTime 문서로 처리되어야 하는가의 여부를 판단한다. (상위엘리먼트 속성의 HyDoc과 각 엘리먼트의 HyTime 키워드로 처리할 정보 검색)

(4) HyTime 구조형태에 해당하는 모든 엘리먼트의 선언이 HyTime표준에 적합한가를 검사한다.

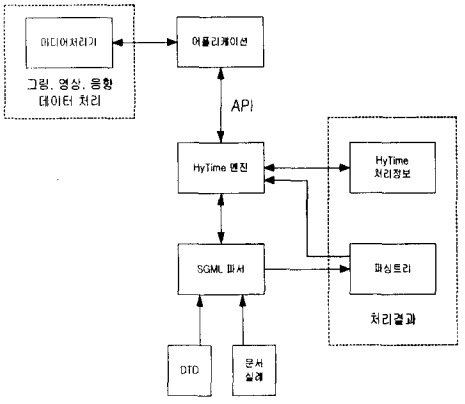
(5) HyQ와 같은 질의에 의한 어드레싱을 위해서 DTD와 DI의 파싱결과를 통해서 해당 엘리먼트의 위치를 찾는 작업이 수행된다.

(6) Dimspec내에서 정의되는 축의 범위가 정해진 제한을 넘어서는가를 검사하고 필요시 링크의 목적 엘리먼트 ID를 반환한다.

마지막 단계에서 앵커들의 정보를 저장하는 이유는 링크될때마다 트리를 추적하는 것 보다 시스템 처리속도를 높일 수 있기 때문이다. 어플리케이션은 처리 정보를 토대로 문서를 표현하게 된다.

6. HyTime 기반 시스템의 구성

HyTime 엔진은 HyTime 기반 시스템의 핵심이 되는 구성요소이다. [그림 3]은 HyTime 엔진이 하이퍼미디어 문서처리 시스템에서의 역할을 도식적으로 나타내고 있다.



[그림 3] HyTime 엔진을 사용하는 시스템의 구성도

하이퍼미디어 시스템은 SGML문서 파싱을 통한 트리정보로 HyTime 엔진의 작업이 이루어진 후에 HyTime엔진 자체 처리정보를 어플리케이션에 넘긴다. 이때 HyTime 엔진이 제공하는 API는 어플리케이션이 특정 작업을 HyTime엔진에 요구하게 되는 경우 제공된다. HyTime의 어플리케이션은 SGML의 파싱결과와 엔진 처리 정보를 가지고 문서를 브라우징하게 된다. 이때 그래픽이나 비디오, 오디오 정보가 문서에 포함되어 있거나 링크 되어 있다면 이를 처리해주는 미디어 처리기를 통해 모든 데이터가 처리된 후 시스템의 문서표현이 종료된다. 통합된 하부구조로 다양한 기능을 제공하는 다수의 어플리케이션의 개발이 가능해진다.

7. 결론

HyTime 엔진의 시스템 내에서의 역할과 기능 등을 살펴보고 설계함으로써 구현의 방향을 제시하는 것이 본 논문의 목표이다. SGML파서로부터 파싱정보를 받고 문서가 HyTime표준에 적합한지를 검사하고 HyTime 모듈에 따라서 해당 작업들을 수행하는 과정으로 엔진을 구성하였다. 엔진은 기본모듈, 위치지정모듈, 하이퍼링크모듈, 스케쥴링모듈, 측정모듈을 적용한다. HyTime엔진의 구현에 앞서 SGML 파서의 선행연구가 필요하며 SGML 파서의 선택은 이전연구[1]에서 획득한 SGML 파서를 이용한다. 현재 많은 어플리케이션들이 하이퍼미디어 문서의 표준인 HyTime을 적용하고 있는 것은 하이퍼미디어 문서표현의 필요를 말한다. 이는 HyTime을 적용하는 어플리케이션들이 HyTime엔진의 처리를 요구함을 뜻한다. 이에 본 논문에서는 하이퍼미디어 문서의 처리를 위한 Hytime 엔진을 설계하였다. 향후과제로는 HyTime엔진과 어플리케이션간의 API와 이에 따른 미디어처리기의 연구가 필요하다.

참 고 문 헌

[1] 변정섭, 신경희, 김상헌, 유재우, SGML 문서를 위한 DTD 파서, 정보과학회지, 제25권, 제1호, 1998.

[2] Martin Bryan, *SGML An Author's Guide to the Standard General Markup Language*, Addison-Wesley Publishing Company, 1988.

[3] John D. Koegel, Lloyd W. Rutledge, John L. Rutledge, Can Keskin, "HyOctane: A HyTime Engine for an MMIS", ACM Multimedia93, Aug. 1993.

[4] Steven J. Derose, David G. Durand, *Making Hypermedia Work a User's Guide to HyTime*, Kluwer Academic Publishers, 1994.

[5] Steven R. Newcomb, Neill A. Kipp, Victoria T. newcomb, "The HyTime Hypermedia/Time-based Document Structuring Language", ACM, Vol.34, No.11, Nov. 1991.

[6] Lloyd Rutledge, Jacco van Ossenbruggen, Lynda Hardman, Dick C.A. Bulterman, "A Framework for Generating Adaptable Hypermedia Documents", ACM Multimedia97, Nov. 1997.