시간종속적 하이퍼미디어 시스템을 위한 SGML DTD의 설계

김 호 진† · 이 수 진†† · 최 윤 철†††

요 약

하이퍼미디어 시스템을 모델링하기 위해 제공되는 기존의 표준 모델들이나 SGML을 이용하여 설계된 하이퍼미디어 시스템들은 저마다 서로 다른 정보 기술 및 저장 방법론을 제시하고 있어 이질적인 시스템간의 문서 교환이나 프리젠테이션을 어렵게 만들고 있다. 그리고 사용자 인터랙션의 처리나 시간종속적 멀티미디어 객체(time-dependent multimedia object)들의 표현에 있어서도 많은 제한점들을 가진다.

본 논문에서는 시간종속적 멀티미디어 객체들을 효율적으로 표현할 수 있고 사용자의 인터랙션에 따라 멀티미디어 객체들을 서로 다른 정보 영역으로 링크시키며 시간종속적 링크를 제공해 주는 시간종속적 하이퍼미디어 시스템(Time-dependent Hypermedia System)을 모델링하고, 그러한 하이퍼미디어 시스템을 기술하기 위한 SGML 문서타입 정의부(Document Type Definition : DTD)를 설계한다.

DTD의 설계에 있어서는 멀티미디어 객체들의 동기화를 위한 스케줄링 메커니즘과 다양한 노드 형태의 기술에 중점을 두고, 사용자들에게 구조 및 개념 파악의 용이성을 제공하기 위해 모듈 개념을 적용하였다.

Design of SGML DTD for Time-dependent Hypermedia Documents

Ho-Jin Kim† · Soo-Jin Lee†† · Yoon-Chul Choy†††

ABSTRACT

Up to the present, numerous standard models have been presented for modeling hypermedia system and these have been used in designing hypermedia systems. However, each of the models and hypermedia systems present a different methods for information representation and storage. This causes a difficulty in document exchange and presentation between different systems. Also, it has many restriction in controlling user interaction and representing time-dependent multimedia objects.

In this paper, a time dependent hypermedia system is modeled. This system can efficiently represent time-dependent multimeia objects and provides time dependent links between multimedia objects in different information space through user interaction. Also, an SGML DTD(Document Type Definition) for hypermedia system is designed.

In designing of DTD, the main focus is scheduling mechanism for synchronization of multimedia objects and description of various node type. Also modular design is applied to facilitate the understanding of the structure and concept of DTD for the users.

† 정 회 원 : 육군본부 C4I개발단
†† 정 회 원 : 육군교육사령부 전산개발처C4I개발단 기반체계처장
††† 정 회 원 : 연세대학교 컴퓨터과학과 교수
논문접수 : 1997년 12월 31일, 심사완료 : 1998년 7월 15일

1. 서 론

1.1 연구 배경

컴퓨터의 발달과 더불어 지금까지는 종이에 작성되던 문서(document)들이 점차 전자 문서화되어 가고 있다. 그리고 멀티미디어 시대의 도래로 이제 문서에는 텍스트 정보만이 아닌 이미지, 동화상, 사운드 등 다양한 미디어 정보들까지도 포함되고 있으며, 여기에 하이퍼텍스트 개념이 도입된 하이퍼미디어 문서가 저작도구 등의 하이퍼미디어 시스템에서 사용되고 있는 상황이다.

그러나 최근 개발된 대부분의 하이퍼미디어 시스템들은 저마다 서로 다른 정보 기술 및 저장 방법론을 제시하고 있으며, 그러한 정보 기술 및 저장 방법상의 다양성은 문서의 교환이나 프리젠테이션을 어렵게 만들고 있다[1].

또한 기존의 하이퍼미디어 시스템들은 링크 메카니즘의 기준이 되는 노드의 형태가 다양하지 못하며, 문서내에 포함된 멀티미디어 객체들의 동기화를 위한 스케줄링이나 사용자 인터랙션의 처리를 제대로 지원하지 못한다.

그럼으로 하이퍼미디어 문서의 기술 및 저장을 위한 표준 포맷은 필수적이라 할 수 있으며 다양한 형태의 노드를 지원하고, 사용자의 인터랙션을 처리하며 멀티미디어 데이타들의 동기화를 위한 스케줄링 메카니즘을 효과적으로 기술할 수 있는 새로운 개념의 하이퍼미디어 시스템에 대한 정의 또한 필요하다.

1.2 하이퍼미디어 문서와 SGML

하이퍼미디어는 멀티미디어와 하이퍼텍스트의 통합적인 확장이다. 멀티미디어는 정보를 표현하는데 있어서 다양한 종류의 데이타를 사용할 수 있도록 해 준다. 반면 하이퍼텍스트는 정보의 순차적 접근방식을 탈피하고 비순차적 접근을 위한 개념적 모델을 제시한다. 그러나 정적정보들에만 적용되던 그 개념은 동적 정보의 복잡한 구성에는 잘 맞지 않는다[2].

일반적인 하이퍼미디어 모델은 전통적인 링크 기반 탐색 항해를 위한 요소와 멀티미디어 프리젠테이션에 필요한 복잡한 타이밍, 그리고 프리젠테이션되는 요소들간의 관계를 기술할 수 있어야한다[Hardman94].

SGML은 문서의 표현과 저장을 위한 표준으로서 문서의 교환을 용이하게 만들며 공통된 하부 구조의 개발을 가능하게 해 준다. 최근에는 이러한 SGML에 근거하여 HyTime이나 TEI 와 같은 여러 응용분야들이 연구되고 있지만 아직은 하이퍼미디어를 포괄적으로 기술할 수 있는 단계는 아니다.

본 논문에서는 하이퍼미디어를 효과적으로 기술하기 위한 SGML응용 시스템 개발의 일환으로 하이퍼미디어 문서의 기술과 교환을 효과적으로 지원해 줄 수 있는 SGML DTD(Document Type Definition: 문서 타입 정의부,이후 DTD)를 설계한다. 하이퍼미디어 문서들마다 가질 수 있는 공통 개념을 분석하여 다양한 하이퍼미디어 문서에 적용될 수 있는 DTD를 설계함으로써 효율적인 멀티미디어 정보의 기술과 네트워크 환경하에서의 문서교환을 지원하고, 특히 시간적 속성을 가지고 있는 멀티미디어 객체들의 표현과 프리젠테이션, 사용자 인터랙션의 처리 등과 같은 사항에 중점을 둔다.

2. SGML

2.1 마크업

마크업이란 문서의 형태를 유지하기 위해 부가적으로 필요한 정보(예를 들면 글자꼴, 글자 크기 등)를 처리하기 위하여 사용되는 명령을 말한다. 마크업은 사용되는 방식에 따라 절차적 마크업과 일반적 마크업으로 나뉠 수 있다.

절차적 마크업은 문서의 일부 내용을 다른 부분과 구분하기 위해 그 모양새를 달리하는 마크업으로 문서의 논리적인 구조를 나타낼 수는 없다. 따라서 서로 다른 응용 시스템간의 호환 유지를 위해서는 마크업을 새로운 시스템에 맞게 모두 바꾸어야 한다. 일반적 마크업은 문서의 논리적 구조 및 속성을 표현하는 마크업으로, 서로 다른 기종간에도 호환 유지가 가능하다.

2.2 SGML 의 구성 요소

SGML이란 마크업 언어가 아니라 마크업 언어의 구문(syntax)을 정의하는 메타 언어이다. 즉, ISO 8879-1986에서 정의된 규칙들을 이용해서 마크업 언어의 구조를 정의하고, 그 정의된 구조에 따르는 마크업 언어를 이용하여 실제적인 문서에 마크업을 하는 것이다[3], [4], [5].

이 SGML 문서는 크게 다음과 같은 세부분으로 이루어져 있다.

◆SGML 선언부
◆문서 타입 정의부(DTD)
◆실제적인 문서(Document Instance : 이후 DI)

2.2.1 SGML 선언부

선언부에서는 SGML 문서에서 어떠한 문자 집합을 이용하는가와 이 문서에서 특수한 기능을 하는 문자에 대한 설명 그리고 이 문서에서 사용되는 기능 등을 알려 준다.

문서 타입 정의부에서는 선언부에서 정의된 문자집합과 기능을 이용하여 문서의 구조를 정의하며, 여기서 정의된 구조를 이용하여 실제적인 문서에 마크업을 한다. 그래서 실제적인 문서는 문서 타입 정의부에 정의된 구조를 따르게 된다. 보통 문서 선언부가 필요한 이유는 문서가 한 시스템에서 다른 시스템으로 전송되었을 경우 시스템들간에 사용되는 문자 코드가 같아야 하고, 특별한 역할을 하는 코드들에 대한 해석이 같아야 한다. 이러한 정보가 선언부에 포함되어 있다.

2.2.2 문서 타입 정의부(DTD)

프로그래밍 언어를 정의하기 위해 BNF - Notation을 이용하는 것처럼 DTD는 하나의 일반적 마크업 언어를 정의한다. 이 때 SGML에 정의된 형식에 맞춰 작성해야함은 물론이다.

문서 타입 정의부는 다음과 같은 세가지 구성요소를 이용하여 문서의 논리적인 구조를 정의한다.

◆엘리먼트(element)

문서의 논리적인 단위를 나타내며, 태그를 통하여 문서의 실제 부분을 마크업한다. 한 엘리먼트는 다른 엘리먼트나 실제 문서를 포함하며, 전체 문서는 엘리먼트를 노드로 하는 트리의 구조를 갖게 된다.

◆엔티티(entity)

엔티티는 ISO 8879에서 "하나의 단위로서 참조되는 문자들의 집합" 으로 정의되어 있다. 즉, 문서의 한 부분 또는 여러 부분에서 그 명칭으로 참조될 수 있는 텍스트를 말한다. 엔티티는 마치 매크로(macro)와 유사하게 사용되며, 내부 엔티티와 외부 엔티티로 나누어 진다.

◆속성(attribute)

엘리먼트의 시작 태그를 수식하는 일종의 매개변수 역할을 하며, 엘리먼트에 부가적인 정보를 나타낸다.

2.2.3 실제 문서(DI)

앞에서 정의한 문서 타입 정의부를 가지고 실제적인 마크업이 삽입되는 부분으로, 실제적인 문서의 내용을 가지고 있다. 여기서 마크업을 할 때 일일이 타이핑을 하여 입력하면 사용자가 번거러울뿐 아니라 오류가 발생하기 쉽다. 그래서 대그 생략 기능이나 태그를 축약(Minimization)하여 쓸 수 있는 기능을 제공한다.

2.3 SGML 파서

파서는 DTD가 ISO표준을 따르는가를 검사하며, 문서가 표현될 수 있는 규칙을 나타내는 문법을 가지고 실제 문서가 그 규칙을 따르는가를 검사하며, 그 결과를 어떤 형식으로든 나타내주는 역할을 한다.

본 시스템에서는 개발의 편의성을 고려하여, 일반적으로 개발이 난이한 것으로 알려진 파서 개발을 피하고 (특히, 고정된 문법을 가진 언어만을 다루는 것이 아닌 경우는 더욱 그렇다), 공개 소프트웨어인 MS-DOS용 파서 ArcSGML을 다소 수정하여 이용하였다.

3. 관련 연구

이 장에서는 본 논문에서 제시하는 DTD에서 참조하고 있는 덱스터 모델, HyTime, HTML에 대해서 구조 형태, 사용자 인터랙션의 처리 여부, 멀티미디어 객체들의 동기화를 위한 스케줄링 메카니즘 등과 같은 사항들에 중점을 두고 각각의 문제점과 특성을 비교, 분석하도록 하겠다. 그리고 분석된 결과를 바탕으로 4장에서는 본 논문에서 제시하는 SGML DTD의 설계에 적용할 새로운 개념의 하이퍼미디어 시스템을 모델링해 보도록 하겠다.

3.1 덱스터 모델(Dexter Model)

덱스터 모델은 하이퍼텍스트의 표준 용어 및 다양한 하이퍼텍스트 모델에서 일반적으로 찾아볼 수 있는 특징들을 추상화한 일반적인 모델을 제공한다. 따라서 덱스터 모델은 여러 하이퍼텍스트 시스템들의 특징과 기능을 비교하고 대조하는 표준으로 사용된다[6].

실질적으로 덱스터 모델은 구체적인 DTD가 존재하는 SGML 어플리케이션은 아니지만 다양한 하이퍼텍스트 시스템의 공통 요소를 제시하고 있기에 새로운

개념의 하이퍼미디어 시스템을 모델링하기 위한 참조 모델로서 사용하고자 한다.

3.1.1 덱스터 모델의 목적

덱스터 모델의 목적은 기존의 하이퍼텍스트 시스템들을 비교하고, 교환 및 상호 작동의 표준을 개발하는데 사용될 수 있는 표준을 제공하는 것이며 다음과 같은 분야에 초점을 맞추어 하이퍼텍스트의 일반적인 모델을 정의하고 있다.

- 저장 계층
- 앵커링 메카니즘
- 계층간의 인터페이스를 형성하는 프리젠테이션 spec

3.1.2 덱스터 모델의 구조

덱스터 모델은 다음과 같은 세 가지 계층(layer)으로 구성되어 있다.

◆Storage Layer

하이퍼텍스트의 핵심인 노드와 링크들간의 네트워크를 기술하며 덱스터 모델의 핵심이다.

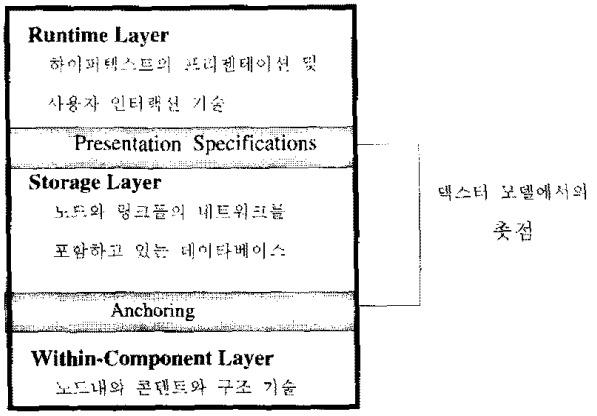
◆Runtime Layer

사용자와 시스템간의 상호작용을 지원하기 위한 메카니즘을 기술한다.

◆Within component Layer

하이퍼텍스트 노드내의 콘텐트와 구조를 다룬다.

위의 세 계층을 도식하면 (그림 3)과 같이 나타낼 수 있다.



(그림 1) 덱스터 모델의 구조
(Fig. 1) Hierarchy of Dexter Model

덱스터 모델에서는 혼란을 피하기 위하여 node라는 용어의 사용을 회피하고 컴퍼넌트(component)라는 중립적인 용어를 사용하여 데이타 모델을 정의한다. 컴퍼넌트는 노드와 링크를 포함하는 콘테이너의 역할을 하며, 컴퍼넌트를 중심으로 어드레싱과 Add, Delete 등의 연산이 이루어진다.

3.1.3 덱스터 모델의 특징

본 논문에서는 하이퍼미디어 시스템을 모델링하고 DTD를 설계하는데 있어 덱스터 모델을 참조하고 있기는 하지만, 덱스터 모델은 텍스트를 기반으로 한 하이퍼텍스트 시스템의 공통적인 요소를 추상화한 모델로서, 다양한 멀티미디어 객체를 다루어야 하는 하이퍼미디어 시스템에 적용하기에는 제한 사항이 많다. 특히 (그림 1)에 나타나 있는 계층들중에서 Within-Component layer는 전혀 다루지 않고, Runtime Layer에서도 프리젠테이션과 사용자 인터랙션 처리에 필요한 가장기본적인 개념만을 제공하고 있으며 계층들을 연결해주는 두개의 인터페이스들도 구체적으로 정의되어 있지는 않다. 그리고 덱스터 모델의 가장 큰 단점은 미디어들의 시간 속성을 전혀 다루지 않는다는 점이다. 따라서 본 논문에서 중점적으로 다루고자 하는 시간종속적 멀티미디어 정보(time-dependent multimedia information)에 대한 프리젠테이션 속성의 처리나 사용자 인터랙션의 처리등이 덱스터 모델에는 전혀 포함되어 있지 않다. 또한 덱스터 모델에서는 컴퍼넌트라는 콘테이너내에 노드와 링크만을 포함시키는 한정적인 개념으로 데이타 모델을 정의하고 있지만 본 논문에서는 그러한 개념을 다소 확장하여 컴퍼넌트가 멀티미디어 객체들의 레이아웃(layout)과 시간적 속성에 관한 정보까지도 모두 포함하는 개념으로 이 후4장에서 제시할 새로운 하이퍼미디어 시스템의 모델링에 적용하였다.

3.2 HyTime (Hypermedia/Time-Based Structuring Language)

1992년 4월 ISO는 HyTime를 국제 표준으로 정하였다. HyTime은 하이퍼미디어 문서들마다 가질 수 있는 공통개념을 묶어 제정된 SGML의 응용분야로서, 하이퍼미디어 문서와 시간성을 기반으로 구조화된 문서의 표현을 위해 개발되었다.

SGML의 중요한 성질중 하나는 마크업이 문서의

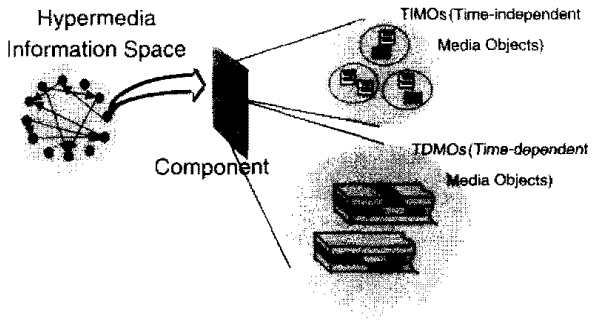
링한 것으로서, 시간종속적 멀티미디어(time dependent multimedia) 정보의 기술에 있어서는 많은 한계를 가진다. 이를 해결하기 위해 본 논문에서는 기존 모델의 문제점을 보완한 새로운 모델을 적용하였다. 본 모델은 고전적인 하이퍼미디어 시스템의 기능과 함께 시간종속 멀티미디어 정보의 프리젠테이션(presentation), 시간종속적 링킹(time-dependent linking), 그리고 사용자 인터랙션(user interaction)을 위한 기능을 가진다.

대부분의 하이퍼미디어 시스템에서 사용자에게 제공되는 정보는 일정한 가공 및 저작 과정을 거쳐 형성된 것이다. 한 시스템내에서 사용되는 모든 정보는 자신을 대표하는 대표자와 실질적인 정보내용 및 다른 정보와의 상관관계를 가진다. 이러한 정보들과 가공 및 저작 과정을 통해 형성된 부가적인 정보들은 시스템 내에서 하나의 정보공간을 형성한다. 모든 정보 시스템은 독자적인 정보공간을 가지고 있으며, 이러한 정보공간의 구성방식은 정보 시스템의 특성과 성능을 좌우하게 된다. 그럼으로 앞에서 시간종속 멀티미디어 정보를 다룰수 있는 시스템을 구성하기 위해서는 우선 적합한 정보공간을 구성할 수 있는 모델이 요구된다.

본 논문에서는 이러한 목적을 위하여 시간종속적 멀티미디어 정보를 효과적으로 다룰 수 있는 시간종속적 하이퍼미디어 정보공간(Time-dependent Hypermedia Information Space)을 제시하고자 한다.

4.1 시간종속적 하이퍼미디어 정보공간

일반적인 하이퍼미디어 시스템에서의 정보공간은 노드와 링크의 네트워크로 구성된다. 본 모델은 노드에 해당하는 요소로 컴퍼넌트(Component)를 사용한다. 컴퍼넌트는 멀티미디어 정보를 시간 및 공간축상에 배치함으로써 형성된 새로운 정보요소로서 동일한 축상에서 사용될 멀티미디어 정보들과 멀티미디어 정보들의 레이아웃(layout) 및 동기화를 위한 스케줄링 정보를 가진다. (그림 2)에서 볼 수 있듯이 한 컴퍼넌트는 두 가지 종류의 멀티미디어 정보들로 구성된다. 각각의 멀티미디어 정보들은 미디어 정보들로 구성된다. 이 때 멀티미디어 정보를 형성 하는 미디어들이 시간축을 가지는가의 여부가 두 멀티미디어 정보의 성격을 구분하는 기준이된다. 시간축을 가지지 않는 멀티미디어 정보는 시간과 관련된 속성의 영향을 전혀 받지 않음으로 이들을 시간독립 멀티미디어 객체(TIMO:Time-independent Multimedia Object)라 하며, 시간축을 가질 경우 시간종속 멀티미디어 객체(TDMO:Time-dependent Multimedia Object)라 한다. 결국 한 컴퍼넌트는 TIMO와 TDMO들의 집합으로 형성된다.



(그림 2) 시간종속적 하이퍼미디어 정보공간
(Fig. 2) Time-dependent Hypermedia Information Space

하이퍼미디어 시스템에서 중심적인 역할을 하는 또 다른 요소는 링크이다. 본 모델에서 링크는 이벤터(Eventor)라는 요소를 통해 구현된다. 이벤터는 프리젠테이션할 미디어를 지정하고 지정된 미디어의 프리젠테이션 속성을 정의하는 역할을 한다. 본 모델에서 이벤터는 또 다른 종류의 미디어일 수도 있으며 혹은 미디어 내부의 내용일 수도 있다. 그럼으로 이벤터는 그 이벤터를 포함하는 멀티미디어 객체가 활성화되어 있을 때에만 나타나며 대응되는 사용자 인터랙션이 발생할 경우에 활성화된다.

결국 본 모델에서 정보공간은 컴퍼넌트의 집합으로 구성되며, 각 컴퍼넌트간의 링크는 컴퍼넌트내에 잠재해있는 이벤터에 의해 형성된다.

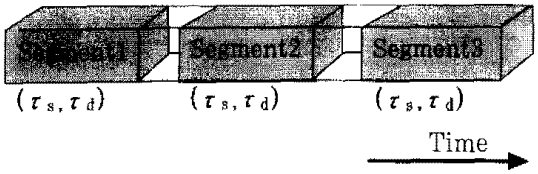
4.2 정보공간의 구성요소

정보공간은 위에서 언급했듯이 크게 두 종류의 요소로 구성되어있다. 하나는 노드를 대표하는 컴포넌트이며, 다른 하나는 링크의 성격을 가진 이벤터이다. 본 절에서는 이들 각각에 대해 언급하고자 한다.

4.2.1 컴포넌트(Component)

컴퍼넌트는 멀티미디어 객체의 집합이다. 그러므로 n개의 멀티미디어 객체에 대해 한 컴퍼넌트는 n-차원 벡터 $\varepsilon = (mmo_1, mmo_2, \ldots\ldots, mmo_n)$ 으로 나타낼

수 있다. 이때 각 멀티미디어 객체는 미디어 객체로 구성되어 있으며 미디어 객체는 다시 세그먼트로 구성된다. 세그먼트는 각 미디어 객체의 부분정보를 가진 단위 로서 시작시간과 지속시간에 대한 정보를 가진다. 결국 미디어 객체는 이러한 세그먼트의 순서집합(ordered set)이다. (그림 3)은 미디어 객체의 구성 방식을 보여주고 있다.



(그림 3) 미디어 객체의 구조
(Fig. 3) Structure of the media object

멀티미디어 객체는 위에서 언급한 미디어 객체들의 집합으로 형성된다. 멀티미디어 객체는 다시 시간종속적인 정보와 시간독립적인 정보로 구분될 수 있다. 시간독립 멀티미디어 객체(TIMO:Time-independent multimedia object)의 경우 미디어 객체의 프리젠테이션이 시간축과 관계없이 이루어진다. 그럼으로 각 미디어간의 스케줄링 정보나 시간축에 관련된 정보를 필요로 하지 않는다. 반면에 시간종속적 멀티미디어 객체(TDMO:Time-dependent multimedia object)의 경우 각 미디어간의 동기화를 위한 정보가 요구된다. 그럼으로 TDMO는 미디어간의 스케줄링 정보와 시간축, 그리고 시간축상으로의 맵핑(mapping)을 위한 정보가 포함된다.

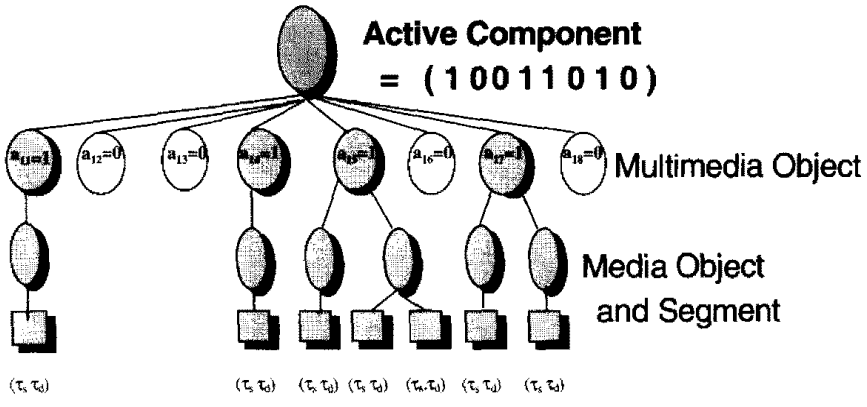
결국 컴퍼넌트는 TDMO와 TIMO라는 두 종류의 멀티미디어 객체들로 구성된 집합이다. 아래 (그림 4)는 일반적인 컴퍼넌트의 구성방식을 보여주고 있다.

4.2.2 이벤터(Eventor)

한 컴퍼넌트내의 모든 멀티미디어 객체가 동시에 사용자에게 제공되는 것은 아니다. 같은 컴퍼넌트내의 멀티미디어 객체라도 이들의 프리젠테이션을 지시하는 지시자에 따라 프리젠테이션될 것인지의 여부가 결정된다. 이러한 역할을 하는것이 이벤터이다. 이벤터는 한 컴퍼넌트내의 멀티미디어 객체중 프리젠테이션될 객체를 지정할 뿐만 아니라 프리젠테이션 방식도 지정할 수 있다.

이벤터는 크게 두가지 종류로 구분되어 있다. 하나는 단순히 프리젠테이션될 컴퍼넌트와 지정된 컴퍼넌트내의 멀티미디어 객체를 지정하는 기능을 수행하는 것이다. 이러한 이벤터를 선택이벤터(Selection Eventor)라고 부른다. 선택이벤터는 인터렉션 종류 μ, 컴퍼넌트 지정자, 그리고 멀티미디어 객체 선택벡터 ε로 구성되어 있으며 다음과 같은 튜플 $<\mu, \rho, \varepsilon>$로 표시할 수 있다. 선택이벤터만으로는 시간종속 멀티미디어 객체의 시간속성을 제어할 수는 없다. 이를 위해 본 모델은 확장형선택이벤터(Extended Selection Eventor)를 동시에 제공한다. 확장형선택이벤터는 프리젠테이션될 미디어의 시간적 위치 λ_s, 지속 시간 λ_d, 그리고 프리젠테이션 속도 ν에 대한 정보가 추가되어 다음과 같은 튜플 $<\mu, \rho, \lambda_s, \lambda_d, \nu, \varepsilon>$로 구성된다.

이벤터를 이용하여 다음과 같은 세가지 기능을 제어할 수 있다.



(그림 4) 활성화된 컴퍼넌트의 일반적인 구조
(Fig. 4) General structure of the active component

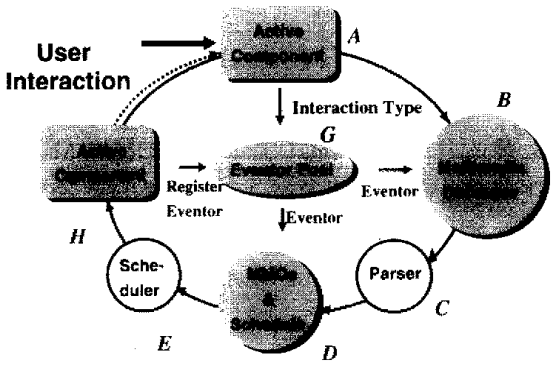
1) 컴퍼넌트 및 멀티미디어 객체의 선택
컴퍼넌트의 선택은 컴퍼넌트 지정자 ρ를 통해 할 수 있다. 선택된 컴퍼넌트 내의 멀티미디어 객체는 멀티미디어 객체 선택벡터에 의해 지정된다. 즉 한 컴퍼넌트내에 멀티미디어 객체가 n개 있고 이 중 첫번째와 마지막 객체만을 활성화할 경우 객체 선택벡터는 (1,0,0,....,1)로 표현된다.

2) 사용자 인터랙션에 의한 프리젠테이션 속성의 조정
한 컴퍼넌트의 프리젠테이션 속도 및 프리젠테이션되는 멀티미디어 객체의 종류를 바꿀 수 있다. 프리젠테이션 속도의 조정은 확장형선택이벤터의 ν를 이용하여 수행할 수 있다.

3) 시간종속링킹
이벤터는 프리젠테이션될 컴퍼넌트를 지정할 수 있을 뿐만 아니라 프리젠테이션될 구간을 지정할 수 있다. 또한 동일한 미디어내에서 시간에 따라 다른 이벤터가 활성화될 수도 있다. 이러한 속성들을 이용하여 결국 시간에 따라 링크될 정보가 계속적으로 변화하는 상황에 응용하여 적용할 수 있다.

4.3 하이퍼미디어 정보공간에서의 프로세싱 방식

컴퍼넌트와 이벤터는 하이퍼미디어 정보공간에서 다음과 같은 방식으로 관리 및 수행된다. 수행방식에 대한 설명은 (그림 5)를 중심으로 설명한다. 우선 각 이벤터는 컴퍼넌트내의 멀티미디어 객체에 내포되어 있다는 것은 전술한 바와 같다[11].



(그림 5) 컴퍼넌트와 이벤터의 프로세싱 방식
(Fig. 5) Processing of Component and Eventor

각 이벤터는 자신이 포함된 컴퍼넌트의 멀티미디어 객체가 활성화되면 이벤트 풀(HG)에 등록된다. 이때 지정된 사용자 인터랙션이 발생하면 이에 해당되는 이벤터를 이벤터 풀에서 찾아 활성화시킨다(AG). 활성화된 이벤터에 의해 멀티미디어 데이타베이스에서 지정된 컴퍼넌트를 찾아 컴퍼넌트내의 멀티미디어 객체를 등록한 후(BCD) 지정된 멀티미디어 객체만을 활성화한다(DH). 만약 이벤터가 이미 활성화되어 있는 컴퍼넌트에 대한 것이라면 멀티미디어 객체의 등록과정은 생략되며, 다만 프리젠테이션 속성만을 변화시킨다(GD).

위와 같은 프로세싱 방식은 본 모델의 컴퍼넌트와 이벤터에 기초한 것으로서 사용자 인터랙션, 시간종속링킹등이 가능함을 알 수 있다.

5. SGML DTD의 설계

5.1 DTD의 필요성

SGML은 문서구조 지정을 위한 기능을 포함하고 있다. 사용자가 자신의 문서를 분석했을 때 그 사용자는 문서 구조를 분석한 결과를 DTD에서 지정하게 된다. 즉, DTD의 목적은 특정 형태의 문서를 구성하기 위한 규칙을 기술하는 것이다.

각각의 SGML 문서는 문서의 구조를 정의하는 DTD로 시작하며, 항상 문서의 시작 부분에 포함되거나 또는 외부 라이브러리로부터 취해질 수 있는 특정한 DTD에 근거하여 해석된다. 따라서 SGML을 이용한 하이퍼미디어 어플리케이션을 설계하고자 한다면 반드시 DTD의 설계가 선행되어야 할 것이다.

5.2 DTD 설계 목표

이미 전술한 바와 같이 기존의 SGML 어플리케이션에서 사용하는 DTD들은 일반적인 컴퓨터 사용자들이 그 개념이나 구조 형태를 이해하기가 용이하지 않고 DTD를 이용한 문서의 작성 또한 어렵다. 그리고 하이퍼미디어 문서에서 중요한 위치를 차지하는 멀티미디어 객체들의 동기화를 위한 스케줄링 메카니즘이 전혀 DTD내에 포함되어 있지 않거나, 그 적용이 어려운 실정이었다. 따라서 본 논문에서 제시하고자 하는 DTD는 그러한 제한 사항을 보완하고 이미 4장에서 제시한 시간종속적 하이퍼미디어 시스템의 특성을 만족하면서 일반 사용자들에게는 좀 더 친숙한 DTD가 될 수 있도록 다음과 같은 목표에 의거하여 작성되었다.

5.2.1 구조 파악의 용이성 제공

DTD는 일단 개념 및 구조 파악이 용이하고 SGML에 대한 기본적인 지식만 있다면 사용자들이 원하는 어플리케이션에 쉽게 적용할 수 있어야 한다. 본 논문에서는 구조파악의 용이성을 제공하기 위해 우선 DTD를 모듈별로 구분하였으며, 하이퍼미디어 문서를 기술할 수 있는 범위내에서 가능한 한 엘리먼트의 수를 줄여 DTD의 크기를 최소화하였다.

5.2.2 멀티미디어 객체들의 동기화를 위한 메카니즘 제공

이미 존재하는 여러 DTD들이 하이퍼미디어 전반을 기술하는 데 있어 특히 문제시되는 부분이 바로 멀티미디어 객체들의 시간/공간적 스케줄링에 관련된 부분이다. HyTime에서는 그러한 메카니즘을 상당 부분 제공하고는 있지만 본 논문의 4장에서 제시하는 시간종속적 하이퍼미디어 모델에 적용하기에는 다소 부족한 부분이 있다. 따라서 본 논문에서 제시하는 DTD는 그러한 부분을 보완하여 일반 사용자들이 하이퍼미디어 문서를 구성함에 있어 멀티미디어 객체들의 시간/공간적 동기화를 위한 스케줄링을 기술하는데 부족함이 없도록 설계하였다.

5.2.3 노드 형태의 다양화

이미 3장에서도 지적했듯이 기존의 SGML 어플리케이션들에서는 정보 구성 요소들간의 링크 메카니즘에 있어 앵커링의 기준이 되는 노드들이 단일 미디어 객체이거나 또는 극히 한정적인 형태로만 지원되었다. 그러나 본 논문에서 제시하고자 하는 DTD에서는 앵커링의 기준이 되는 노드를 단일 미디어 객체의 일부분에서부터 하나의 하이퍼미디어 문서에 이르기까지 여러 형태로 구분하여 다양한 노드들이 존재할 수 있도록 하였다.

5.2.4 사용자 인터랙션의 처리

HTML을 제외한 대부분의 SGML 어플리케이션들이 시스템과 사용자간의 상호작용(interaction), 즉 사용자 인터랙션을 제대로 처리하지 못하고 있다. 그리고 HTML의 경우에도 한정적인 부분에 한해 사용자의 입력을 처리할 수 있는 요소들을 제공하고 있다. 따라서 본 논문에서는 다양한 사용자의 입력과 시스템과의 상호작용을 처리할 수 있는 요소들을 포함하여 설계하였다.

5.3 DTD의 구성

5.3.1 DTD 모듈

앞장에서 소개한 HyTime에서는 메타 DTD를 6개의 모듈로 나누어 설계하였다. 본 논문에서 제시하는 DTD도 차후 사용이나 개념 파악의 용이성을 위해 모듈 개념을 도입하여 적용하였으며 다음과 같이 Entity Definition Module을 포함하여 총 6개의 모듈로 구성되어 있다.

- ◆Entity Definition Module
- ◆Base Module
- ◆User Interaction Module
- ◆Location Address Module
- ◆Hyperlink Module
- ◆Scheduling Module

5.3.2 Entity Definition Module

대부분의 DTD들은 엔티티 정의를 포함한다. 엔티티 정의는 매크로(macro) 정의와 유사하며 DTD를 작성할 때나 또는 인스탄스에 태그를 붙일 때 키입력을 줄여준다. 엔티티 정의가 다른 모듈정의에 앞서 처음 부분에 나온 이유는 DTD의 파싱 과정에서 사용하는 ArcSGML의 특성상 엔티티 참조 이전에 엔티티가 선언이되어 있어야 하기 때문이다.

```
<!ENTITY % InputType "(TEXT|
 PASSWORD|CHECKBOX|RADIO)">
```

5.3.3 Base Module

텍스트 엘리먼트인 HyMedia를 정의하며, 하이퍼미디어 문서의 내용과 직접적 으로 연관된 요소들의 대부분을 본 모듈에서 정의하고 있다. 그리고 Base Module에서는 앵커링의 기준이 되는 노드들을 다양한 형태로 정의하고 있다.

HyMedia

텍스트 엘리먼트로서 트리 구조상의 가장 상위 노드에 해당하며 front, body, back으로 구성되어 있다.

① front

HyMedia의 전체적인 제목과 문서에 대한 정보를 가지고 있는 엘리먼트foreword와 객체들에 대한 계층적인 정보(global map & local map)를 가지고 있는

mapinfo엘리먼트로 구성되어 있다.

② body

하이퍼미디어 문서에 포함되는 요소들, 즉 실질적인 미디어 객체들이 정의되어 있으며, 그러한 미디어 객체들이 조합되어 구성되는 다양한 노드들도 함께 정의하고 있다. body에서 정의하고 있는 노드들의 연관관계는 (그림 6)과 같다.

● component

body를 구성하고 있는 주 엘리먼트로서, 텍스터 모델의 컴퍼넌트(component) 개념을 도입하여 사용하였다. 각각의 컴퍼넌트는 멀티미디어 객체인 mmo들의 집합으로 이루어진다.

● mmo (Multimedia Object)

mmo는 다시 동기화 메카니즘이 적용되는지의 여부에 따라 TIMO와 TDMO로 나누어진다.

● TIMO (Time Independent Multimedia Object)

시간성을 가지지 않고 독립적으로 프리젠테이션되는 텍스트와 그 안에 포함된 이미지 또는 사용자의 인터랙션에 의하여 다른 미디어 객체들과는 무관하게 프리젠테이션되는 객체들의 집합을 표현하기 위한 엘리먼트이다.

● TDMO (Time Dependent Multimedia Object)

미디어 객체들간에 밀접한 동기화가 요구되는 경우, 그런 객체들로 이루어진 집합을 기술하기 위한 엘리먼트로서 TIMO와는 달리 동기화에 필요한 정보를 미디어 객체와 함께 가지고 있다.

● mo

여러개의 미디어 객체들의 조각들이 모여 구성된 단위로서 동기화와 관련된 정보들은 가지고 있지 않다.

● segment

단일 미디어 객체의 일부분으로 구성된 노드를 기술하기 위한 엘리먼트이다

● mcontent

단일 미디어 데이타들을 기술하기 위한 엘리먼트로서 다음과 같은 요소들을 포함한다.

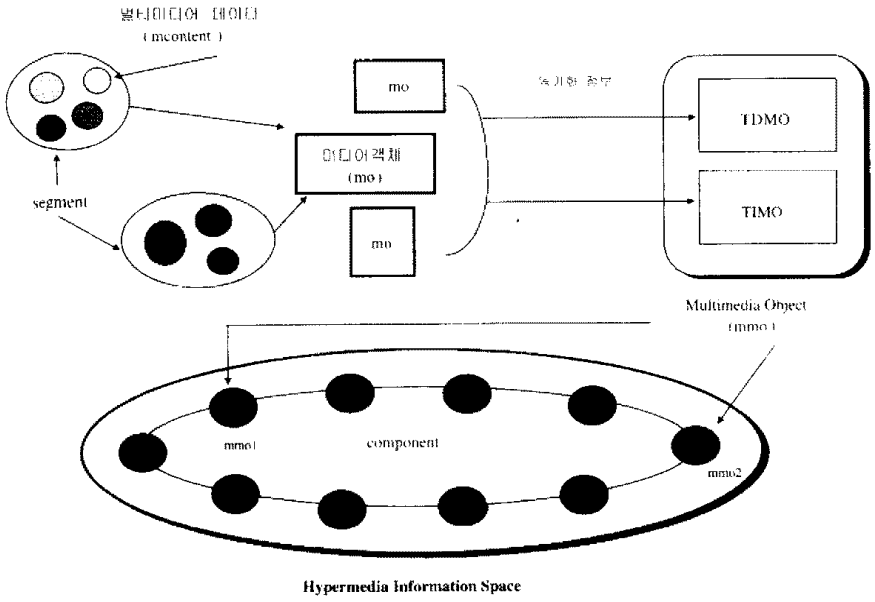
(Text | Image | Sound | Music | Video | Animation | Script | Program)

● back

작성된 하이퍼미디어 문서에 관한 인덱스나 부록 등을 기술하는 부분으로 <appendix>, <index>와 같은 요소들로 구성되어 있다.

5.3.4 User Interaction Module

사용자의 입력이나 하이퍼미디어 시스템과의 상호



(그림 6) 노드들의 연관관계
(Fig. 6) Relation between nodes

작용(interaction)을 처리하기 위한 모듈로서 다음과 같은 요소들이 정의되어 있다.

● input

작성된 하이퍼미디어 문서의 프리젠테이션시 사용자에게 텍스트의 추가적인 삽입이나 또는 CHECKBOX, RADIO버튼을 통한 선택사항의 입력이 필요할 경우가 있다. 그러한 경우를 위하여 본 모듈에서는 사용자의 입력을 처리할 수 있는 input 엘리먼트를 정의하고 있다.

```
<!ELEMENT  input        - o        EMPTY>
<!ATTLIST  input
           type         %InputType  "TEXT"
           value        CDATA       #IMPLIED
           checked      (checked)   #IMPLIED
           size         NUMBER      #IMPLIED
           maxlength    NUMBER      #IMPLIED>
```

엔티티 InputType은 엔티티 정의 모듈에서 다음과 같이 정의되어 있다.

```
<!ENTITY  % InputType  "(TEXT |
 PASSWORD | CHECKBOX | RADIO)">
```

그리고 나머지 속성들 중 value는 RADIO버튼과 CHECKBOX의 값을 기술하기 위한 속성으로 "Node Insert", "Volume up" 등의 값을 가질 수 있다. checked 속성은 CHECKBOX와 RADIO 버튼을 위한 속성으로 어떤 값이 사용자에게 선택되었을 때의 on/off상태를 지정하는데 사용된다. 디폴트값은 off이며 속성값이 "checked"로 주어졌을 경우 사용자에 의해 버튼이 선택되었다는 on 상태가 된다. size속성은 Input field의 크기 지정을 위한 속성이며, maxlength는 Input field에 들어가는 입력 텍스트의 최대 길이를 지정하나 일반적으로 size 속성과 같은 값을 가진다.

● button

이미 4장에서 제시한 시간종속적인 하이퍼미디어 시스템에서 사용자의 입력을 처리하기 위한 요소는 필수적이었다. button엘리먼트는 사용자의 입력을 처리하여 이벤터를 활성화시키는 역할을 하며, 단순히 미디어 객체만을 지정하는 이벤터인 seleventor와 미디어 객체에 대한 링크 및 그 미디어 객체들이 재생될 시간에 관한 정보를 모두 가진 이벤터인 exteventor로 이루어진다. exteventor에 대한 설명은 이후에 나오는 Hyperlink Module에서 하도록 하겠다.

5.3.5 Location Address Module

하이퍼미디어 문서에서는 특정 객체에 대한 직접적인 링크외에도 사용자가 원하는 부분에서 특정 객체를 참조하게 될 경우가 발생한다. 이러한 경우를 위해 본 모듈에서는 문서내에서의 객체의 위치 지정을 위한 메카니즘을 제공한다.

* 객체의 위치 지정법 (Object Locating)

문서내에서 사용자가 참조하고자 하는 특정 객체의 위치를 저정하기 위해서는 다음과 같은 세가지 기법 중 어느 한 가지나 또는 두 가지 이상의 기법이 복합적으로 적용될 수 있다. 본 논문에서 제시하는 DTD에서는 마지막 방법인 질의어나 속성에 의한 위치 지정 방법은 지원하지 않는다.

1. 명칭에 의한 위치 지정
2. 카운팅에 의한 위치 지정
3. 질의어 또는 객체의 속성에 의한 위치 지정

5.3.6 Hyperlink Module

앵커와 링크에 관한 사항을 정의함으로써 사용자들이 정의한 문서에 포함된 모든 객체를 참조할 수 있는 메카니즘을 제공하는 모듈이다.

① clink(contextual link)

일반적인 크로스 참조나 항해 링크를 표현하는데 사용되는 링크로서 링크 자체가 나타나는 위치와 다른 위치를 연결한다. 원래의 앵커를 포함하면서 문서내에 저장됨으로 문서가 읽기 전용이거나 접근이 불가능할 경우 링크에 대한 편집이 불가능하게 된다.

```
some topics, like <clink linkend=dest1> links</clink>
are complex.
```

② ilink(independent link)

둘 이상의 위치를 연결하거나 또는 앞에서 소개한 clink와는 달리 편집이 가능하도록 문서와는 별도로 저장되는 링크를 필요로 하는 어플리케이션을 위해 제공하는 링크로서 끝점(endpoint)으로부터 분리되어 저장된다. 즉, 앵커로부터 독립된 문서내에 저장이 가능하다. 그럼으로 읽기 전용 목적의 문서에 주석을 붙이는

대 사용할 수 있다.

```
<p id=nels1> Nelson's notion of "deep rock archives"
. . .
<p id=bush>Vannevar Bush's concern about the
 possibility of nuclear war
. . .
<ilink linkend="nels1 bush">
```

다음에 설명할 두 가지의 엘리먼트 seleventor와 exteventor는 링크와 동일한 역할을 하지만, 이미 4장에서도 설명한 것처럼 단지 링크의 목표만을 지시하던 기존의 링크와는 달리 링크가 지시하는 객체들에 적용되어야 할 정보들까지도 포함하고 있다.

● seleventor(Selection Eventor)

4장에서 전술한 바와 같이 선택 이벤터인 seleventor는 단순히 프리젠테이션된 컴퍼넌트와 지정된 컴퍼넌트내의 멀티미디어 객체를 지정하는 기능을 수행하며 튜플 $<\mu,\rho,\varepsilon>$로 표시할 수 있다고 했다. 그러한 정의에 따라 선택이벤터는 다음과 같은 엘리먼트로 정의할 수 있다. 튜플에서 사용자의 인터랙션 종류를 나타내는 μ는 속성 inttype으로 정의하였으며, 프리젠테이션된 컴퍼넌트를 지정하는 ρ는 destloc 속성으로 정의하여 컴퍼넌트의 id값으로 참조가 이루어지도록 하였다. 그리고 컴퍼넌트내의 멀티미디어 객체를 지정하는 객체 선택 벡터 ε는 참조할 멀티미디어 객체의 id값들을 seleventor의 내용으로 나열함으로써 참조가 이루어지도록 정의하였다. 즉, 활성화시킬 객체의 선택은 그 객체들의 id값들을 명시함으로써 이루어진다.

```
<!ELEMENT seleventor      -  -   (#PCDATA)>
<!ATTLIST seleventor
          id        ID        #IMPLIED
          inttype   NAME      #REQUIRED)
          filename  CDATA     #IMPLIED
          destloc   IDREF     #REQUIRED>
```

● exteventor(Extended Selection Eventor)

선택 이벤터만으로는 시간 종속 멀티미디어 객체의 시간 속성을 제어할 수 없기에 시간에 대한 정보를 추가한 확장형 선택 이벤터를 다음과 같이 정의한다.

```
<!ELEMENT exteventor      -  -   (#PCDATA)>
<!ATTLIST exteventor
          id        ID        #IMPLIED
          inttype   NAME      #REQUIRED
          filename  CDATA     #IMPLIED
          destloc   IDREF     #REQUIRED
          start     CDATA     #REQUIRED
          duration  CDATA     #REQUIRED
          factor    NUMBER    #REQUIRED
          cooratio  IDREF     #IMPLIED >
```

4장에서 확장형선택이벤터는 선택이벤터에 프리젠테이션된 미디어의 시간적 위치 λ_s와 지속 시간 λ_d, 그리고 미디어의 프리젠테이션 속도 ν에 대한 정보가 추가되어 다음과 같은 튜플 $<\mu,\rho,\varepsilon,\lambda_s,\lambda_d,\nu>$로 구성된다고 했다.

튜플에서 μ,ρ,ε는 선택이벤터와 동일한 의미를 가지며, 시간적 위치를 나타내는 λ_s는 속성 start로, 지속 시간을 나타내는 λ_d는 duration으로, 그리고 프리젠테이션 속도를 나타내는 ν는 factor로 정의하였다. 또한 id값으로 참조하는 속성cooratio는 시간적 정보를 나타내는 속성들인 start, duration, factor 속성들을 지정하지 않고 다음 절에 소개할 엘리먼트 cooratio의 id값을 지정함으로써 시간적 정보들을 참조하여 미디어 객체에 적용할 때 사용하는 속성이다.

5.3.7 Scheduling Module

멀티미디어 객체는 많은 모노 미디어들이 시간적, 공간적 관계를 가지고 결합되어 있는 정보이다. 멀티미디어 시스템은 그러한 멀티미디어 객체들로 구성되어 있다고 볼 수 있다. 기존의 멀티미디어 시스템들은 멀티미디어 객체를 이루는 미디어들에 대해 이미 구성된 시간, 공간적 구조를 반영하여 단순히 재생하는 방식의 프리젠테이션을 제공한다. 그러나 하이퍼미디어 시스템이나 디지탈 라이브러리 등에서는 그러한 단순 재생은 큰 의미를 가지지 못한다. 다시 말해서, 사용자는 멀티미디어 객체를 이루는 모노 미디어를 선택적으로 재생하고 재생 속도를 바꾸거나 또는 현재 재생중인 객체의 재생 구간을 선택할 수 있어야 한다는 말이다.

Scheduling Module은 멀티미디어 객체들간의 시간 및 공간적 동기화를 위한 스케줄링에 필요한 정보를 기술하는 모듈로서 동기화의 표현에 있어서는 절대적

표현과 상대적 표현이 모두 가능하게 해 준다. 그리고 비디오나 사운드 객체의 경우 재생 구간의 선택은 물론 동일한 객체에 대한 재생 속도의 변경까지도 가능하다. 또한 본 모듈은 멀티미디어 객체들의 공간적인 스케줄링을 위한 요소들도 포함하고 있어, 멀티미디어 객체의 상대적인 위치 지정과 배치에도 사용할 수 있다.

본 모듈에서는 다양한 측면에서의 동기화 표현을 지원하기 위하여 요소들을 좀 더 세부적으로 정의하였다. 따라서 정의된 엘리먼트와 속성들을 충분히 잘 활용한다면 사용자의 의도에 따라 여러 형태의 표현 방식이 가능해질 것이다. 그 세부적인 사항들은 DTD와 함께 예를 들어 설명하겠다.

● cooratio (coordination ratio)

미디어 객체에서 부분적인 재생을 위해 재생 구간을 선택한 후 선택된 구간에 대한 재생속도의 변경을 원할 경우 사용하기 위해 정의된 엘리먼트이다.

```
<!ELEMENT cooratio   - o          EMPTY>
<!ATTLIST cooratio
          id         ID           #REQUIRED
          type       CDATA        #REQUIRED
          absunit    CDATA        #REQUIRED
          dimunit    CDATA        #REQUIRED
          ratio      NUMBER       #REQUIRED
          rattype    (RATIO|NONE) "RATIO">
```

위의 정의를 바탕으로 하여 실제 문서에서는 다음과 같이 태그를 붙일 수 있다.

```
<cooratio type = "frame" absunit = 0.1 dimunit
  = "f" ratio = 3 >
```

type속성은 미디어 객체들의 일반적인 재생 단위가 되는 frame(비디오)이나 byte(오디오) 등을 기술하는 속성이며, absunit는 재생속도를 지정하기 위한 단위 시간을 나타낸다. dimunit속성은 frame의 "f"를 표시하며, ratio는 지정된 단위 시간에 몇 frame이나 byte가 플레이되어야 하는지를 지정한다.

● measure

미디어 객체의 일부분인 세그먼트(segment)에 적용될 시간적인 정보를 기술하는 엘리먼트로서 앞에서 정의한 cooratio엘리먼트의 id값으로 세그먼트에 적용할 시간 정보를 참조한다.

● trsch (Temporary Relational Scheduler Element)

멀티미디어 객체에 포함되어 있는 각각의 미디어 데이타들간의 시간적인 관계 즉 미디어들간의 동기화를 위한 엘리먼트로서 절대적인 동기화를 표현하는데 사용되는 엘리먼트 abssch와 7가지의 상대적 관계를 이용하여 상대적인 동기화를 표현하는데 사용하는 엘리먼트 relsch로 나누어진다.

① abssch (Absolute Scheduler Element)

절대 시간의 개념에 의해 절대적인 동기화를 기술하기 위한 엘리먼트로서 time 엘리먼트를 가진다.

② relsch (Relative Scheduler Element)

상대적인 동기화를 표현하기 위한 엘리먼트로서 상호 관계를 가지는 2개의 미디어를 지정하는 엘리먼트 primoseg 와 secmoseg, 그리고 지정된 두 미디어들간의 상대적 관계를 기술하는 엘리먼트 relop로 구성되어 있다.

● primoseg(Primary Media Object Segment Element)

상대적 관계를 표현하기 위해 기준이 되는 미디어 객체의 세그먼트를 나타내며, 세그먼트의 id값으로 참조한다.

● secmoseg(Secondary Media Object Segment Element)

primoseg에 대해 상대적 관계를 가지는 세그먼트를 나타내며, id값으로 참조한다.

● relop (Relative Operation)

(그림 7)과 같은 7가지의 상대적 관계를 이용하여 세그먼트들간의 상호관계를 표현함으로써 동기화를 기술한다[Keogel 94].

```
<!ELEMENT relop    - o          EMPTY>
<!ATTLIST relop
          type     (BEFORE | MEETS | OV
                   ERLAPS |   DURING |
                   STARTS   | FINISHES |
                   EQUALS)
                                "BEFORE"
```

```
        param1    CDATA           #IMPLIED
        param2    CDATA           # IMPLIED
        apply     (PRIMARY | PREVIOUS |
                   ALL)           "PRIMARY">
```

relop 엘리먼트의 속성들은 다음과 같은 의미를 가지며, 그 중 param1과 param2는 상대적 관계의 값에 따라 해석이 달라질 수 있다.

param1

secmoseg가 primoseg에 대해 어느 정도의 시간 간격을 가지고 재생이 시작 되어야 하는지를 지정하는데 사용한다. param1 속성의 값이 지정되는 경우는 상대적 관계가 before, overlaps, during, finishes 일 경우이다.

param2

secmoseg가 primoseg에 비해 어느 정도의 시간 간격을 가지고 재생이 끝나야 하는지를 지정하는데 사용한다. param2속성의 값이 지정되는 경우는 overlaps, during, starts가 될 경우이다.

apply

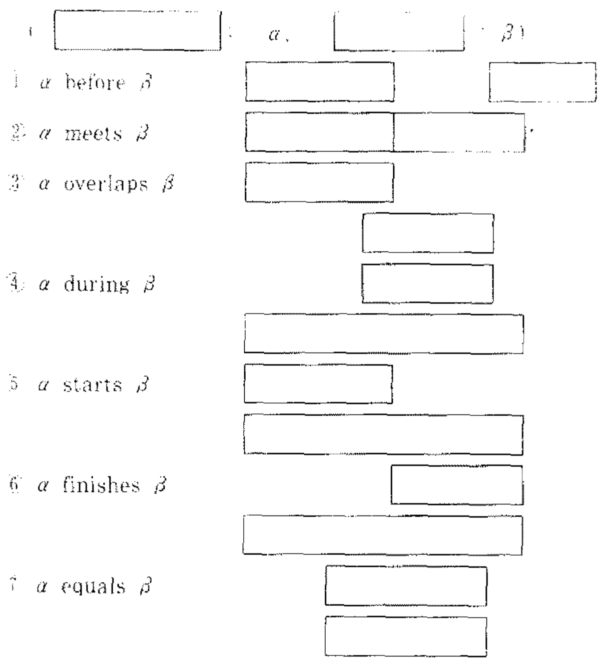
secmoseg에 대해 상대적 관계를 가지는 primoseg를 지정하는데 사용되며, 값으로 "PRIMARY", "PRIVIOUS", "ALL"을 가진다. 그 값이 PRIMARY일 경우에는 맨 처음에 primoseg로 지정한 세그먼트가 되며, PRIVIOUS일 경우에는 바로 이전에 지정되었던 세그먼트가 되고, ALL일 경우에는 이전에 지정되었던 모든 세그먼트에 대해 상대적 관계가 적용됨을 의미한다.

6. 적용 및 분석

본 논문의 DTD는 멀티미디어 객체들간의 동기화를 위한 스케줄링 메카니즘에 중점을 두고 설계되었고, 시간종속적 하이퍼미디어 모델에서 이벤터라는 새로운 링크의 개념을 도입하여 DTD 설계에 적용하였다. 따라서 본 장에서는 그러한 면들에 초점을 맞추어 실제 문서를 작성하고 이론적인 검증을 실시하도록 하겠다.

6.1 적용

다음에 기술되는 예들은 전체적인 DI의 일부분으로서 음성이 지원되는 동화상에서 비디오 프레임과 사운드의 동기화를 기술하는 부분이다.



(그림 7) 7 가지의 Binary Temporal Relations
(Fig. 7) Binary Temporal relations

6.1.1 비디오의 재생

비디오의 재생 예

```
1: <cooratio id="c1" type="frame" absunit= 0.1 dim
     unit="f" ratio=2>
2: <mo id="mediaAa" medium="video">
3:      <measure cratio="c1">
4:      &lt;segment id="segAa1" startloc="1
           f" extend= "30 f">
5:           <mcontent><Video filename="tel.avi">
              </mcontent>
6:      &lt;/segment&gt;
7: </mo>
```

위 예의 2번째 줄을 보면 비디오의 경우 재생할 동화상 파일의 명칭만으로 참조가 이루어진다. 이는 사운드의 경우도 마찬가지이다.

1번째 줄에서는 재생될 비디오 객체의 재생 속도를 기술하고 있다. type속성은 비디오의 프레임을 나타내며, absunit 속성의 값으로 주어진 0.1은 단위 시간을 나타낸다. 속성 dimunit은 값 "f"는 단위 시간당 재생

된 비디오 객체의 단위인 프레임을 의미한다. 그리고 마지막 속성인 ratio는 단위 시간당 재생될 비디오 프레임의 수를 의미한다. 따라서 전체적인 의미는 "tel.avi"라는 동화상 파일을 0.1초당 2프레임의 속도로 재생하라는 의미가 된다. 4번째 줄에서는 그와 같은 재생속도가 적용될 부분이 "tel.avi" 파일의 1프레임에서 30프레임까지임을 의미한다.

6.1.2 사운드의 재생

사운드의 재생 예

```
1: <cooratio id="c2" type="byte" absunit=0.1
   dimunit="b" ratio=512>
2: <mo id-"mediaAb" medium="sound">
3:      <measure cratio-"c2">
4:      <segment id-"segAb1" startloc="1
        b" extend= "10 b">
5:          <mcontent><Sound filename-"tel.wav">
            </mcontent>
6:      </segment>
7: </mo>
```

위의 예에서 나타난 것처럼 사운드의 경우 비디오와 대부분 동일하다. 단지 재생의 단위가 프레임이 아니라 byte라는 것만 차이가 난다.

6.1.3 동기화의 표현

이미 5장에서 설명한 7가지의 관계를 이용하여 비디오와 사운드를 다음의 예와 같이 동기화시킬 수 있다.

```
1: <segment id-"segAa1"  startloc="1 f" extend=
   "30 f">
2:     <mcontent><Video filename="tel.avi">
       </mcontent>
3: </segment>
4: <segment id="segAa2"  startloc-"40 f" extend
   ="50 f">
5:     <mcontent><Video filename="tel.avi">
       </mcontent>
6:  </segment>
7:  . . .
8: <segment id="segAb1" startloc-"1 b" extend=
   "10 b">
9:     <mcontent><Sound filename="tel.wav">
       </mcontent>
10: </segment>
11: . . .
12: <trsch>
13:  <relsch>
14:    <primoseg segid="segAa1">
15:    <relop param1="0" param2="0"
         apply="PRIMARY">
16:    <secmoseg segid="segAb1">
17:    <relop type="EQUALS" param1="0"
         param2="0" apply="PREVIOUS">
18:    <secmoseg segid="segAa2">
19:  </relsch>
20: </trsch>
21: . . .
```

이미 앞에서 비디오와 사운드 객체의 구간 선택 재생과 재생 속도의 변경등에 관하여 알아 보았다. 위의 예에서 12 - 20 번째 줄까지는 그런 선택 사항에 적용할 동기화 메카니즘이 기술된 부분이다. 상대적관계를 나타내는 엘리먼트인 relop의 경우 "MEETS"가 디폴트값으로 정해져 있다.

5장에서 설명한 내용들을 기초로 하여 그림의 내용을 해석하면 다음과 같다. 먼저 14번째 줄에서는 상대적인 관계의 기준이 되는 세그먼트를 지정하고 있으며, 15,16번째 줄에서는 그 세그먼트에 대해 상대적 관계를 가지는 세그먼트와 그들간의 상대적 관계를 기술하고 있다. 그 의미는 "tel.avi" 파일의 1프레임부터 30프레임까지 재생된 후, 40프레임부터 50프레임까지가 연속해서 재생됨을 나타낸다. 17번째 줄에서는 속성 apply의 값이 "PREVIOUS"로 지정되어 있음으로 바로 이전 세그먼트, 즉 16번째 줄의 세그먼트와 상대적 관계를 가짐을 의미한다. 따라서 "tel.avi"파일의 40프레임부터 50프레임까지와 "tel.wav"파일의 1byte부터 10byte까지가 동시에 재생되어 동시에 끝남을 의미한다(단, 비디오 세그먼트와 사운드 세그먼트의 재생 시간은 동일한 것으로 가정한다).

전체적인 의미는 "tel.avi"파일의 1프레임부터 30프레임까지 재생된 후 곧 이어 40프레임부터 50프레임까지가 재생되면서 "tel.wav"파일의 1byte부터 10byte까

시가 동시에 재생됨을 나타낸다.

6.1.3 이벤터의 적용

본 논문의 4장에서 제시하는 시간종속적 하이퍼미디어 시스템에서 이벤터는 중요한 위치를 차지한다.

이벤터는 하이퍼미디어의 링크와 같은 역할을 담당한다. 그러나 기존의 링크 와는 달리 사용자 인터랙션의 처리, 시간종속 링킹 등의 기능을 가질 수 있다. 이벤터는 크게 나누어 다음의 세가지 목적으로 사용될 수 있다. 첫째, 프리젠테이션될 컴퍼넌트 또는 컴퍼넌트내의 멀티미디어 객체의 선택시 사용된다. 둘째, 멀티미디어 객체의 프리센테이션 속성을 변경할 때 사용할 수 있다. 셋째, 시간 종속 링크로서 사용될 수 있다.

그러한 기능을 수행하는 이벤터의 적용 예를 보이기 위해 다음의 예와 같이 컴퍼넌트를 정의한다.

```
<component  id=10>
        ...
    <TDMO id=101>  ...video.avi.....</TDMO>
    <TDMO id=102>  ...audio.wav.....</TDMO>
    <TIMO id=103>
        ...<text>
         <seleventor  inttype=MOUSE_CLICK
         destloc=10>101</seleventor>  PLAY
         VIDEO</text>....
    </TIMO>
    <TIMO id=104>
        ...<text>
         <seleventor  inttype=MOUSE_CLICK
         destloc=10>102</seleventor>  PLAY
         AUDIO</text>....
    </TIMO>
    <TIMO id=105>
        ...<text>
         <exteventor inttype=MOUSE_CLICK
          destloc=10, start= CURRENT,
              duration=UNDEFINE, factor
                  =0>101,102</exteventor>
    PAUSE </text>
    </TIMO>
        ...
</component>
```

윗의 예는 두개의 TDMO를 가진 컴퍼넌트에서 각각의 TDMO를 이벤터를 이용하여 선택하는 것을 보인 것이다. 위의 컴퍼넌트가 활성화되면 우선 TIMO만이 활성화된다. 각 TIMO들은 텍스트로 구성되어 있으며 동시에 이벤터를 내포하고 있다. TIMO(id=103)내의 이벤터는 TIMO의 텍스트 영역내에서 마우스가 클릭되면 TDMO(id=101)를 활성화시킨다. TIMO(id= 105)는 확장형선택이벤터를 가지고 있다. 이 이벤터는 현재의 컴퍼넌트내의 두 TDMO(id=101,102)에 대한 프리센테이션 속도를 0으로 지정하고 있다. 이는 두 TDMO의 활성화를 중지하는 역할을 하게된다. 만약 factor가 1보다 크다면 fast forward의 역할을 하게되며 0과 1사이일 경우에는 slow, 0 보다 작을 경우 backward 효과가 발생하게 된다.

다음은 시간종속 링킹의 예를 보이고자 한다. 아래에는 1개의 비디오 정보로 구성된 TDMO를 포함하고 있는 컴퍼넌트가 기술되어 있다. 이 컴퍼넌트내의 TDMO는 3개의 세그먼트로 구성된 한개의 미디어를 가진다. 이때 각 세그먼트 내에는 서로 다른 선택이벤터가 기술되어 있다.

```
<component id=11>
   <TDMO id=101>
     ...
      <mo>
      ...
          <segment  id=1001  measure="frame"
          startloc=1 extend=1000>...video.avi ...
          <seleventor  inttype=MOUSE_CLICK
          destloc=10>103 104</seleventor>
          </segment>
          <segment  id=1002  measure="frame"
          startloc=1001 extend=1000>...video. avi...
          <seleventor  inttype=MOUSE_CLICK
          destloc=12>101 102</seleventor>
          </segment>
          <segment  id=1003  measure="frame"
          startloc=2001 extend=1000>...video. avi...
          <seleventor  inttype=MOUSE_CLICK
          destloc=13>101 102 103 104 </seleventor>
           </segment>
      </mo>
```

...
</TDMO>
</component>

앞의 컴퍼넌트에서는 첫 세그먼트가 플레이되는 도중에 마우스 클릭이 발생하면 id=10인 컴퍼넌트가 활성화된다. 반면 두번째 및 세번째 세그먼트가 플레이되는 동안 사용자 인터랙션이 들어오면 id가 12인 컴퍼넌트 또는 13인 컴퍼넌트가 활성화된다. 이와 같이 동일한 객체가 시간에 따라 다른 목적영역으로 링크를 형성하게 된다. 이런 개념의 링크는 기존 하이퍼미디어 시스템에서는 지원해 주지 못하고 있다.

6.2 분석

본 논문에서 제시하는 DTD와 그 DTD을 이용하여 작성한 DI의 이론적인 분석을 통해 멀티미디어 객체의 동기화 메카니즘에 관하여 알아보았다. 비록 이론적인 면에서의 고찰이었지만 본 논문에서 제시하는 DTD를 이용하여 작성된 DI에서는 멀티미디어 객체를 구성하는 미디어들간의 동기화를 충분히 표현할 수 있었다. 그리고 각 미디어들에 대하여 재생 구간의 선택은 물론 선택된 구간에 대한 재생 속도의 변경 또한 가능하였다.

또한 기존의 하이퍼미디어 시스템에서는 제공되지 않았던 시간종속적 링크인 이벤터의 도입은 앞의 예에서도 알 수 있듯이 동일한 미디어 객체가 사용자의 인터랙션에 의하여 서로 다른 목적영역으로 링크될 수 있게 함으로써 사용자 인터랙션의 처리는 물론, 링크에 의해 프리젠테이션될 객체의 속성 변경까지도 가능하도록 해 주었다.

7. 결론 및 향후 연구 방향

기존의 SGML을 이용한 어플리케이션에서 대부분의 DTD들은 그 구조형태를 파악하기도 어려울 뿐만 아니라 일반 사용자들이 하이퍼미디어 문서를 작성하기 위해서는 상당한 연구를 요하는 부분들이 많았다. 그리고 대부분의 하이퍼미디어 시스템에서는 시간과 공간적 관계를 가지고 상호 밀접하게 연결된 멀티미디어 객체들의 동기화를 위한 스케줄링 메카니즘에 있어 메카니즘을 구현하기 보다는 멀티미디어 객체를 이루는 미디어들에 이미 구성된 시간, 공간적 구조를 반영하여 단순히 재생하는 방식의 프리젠테이션을 제공하거나 또는 그러한 메카니즘이 전혀 기술되어 있지 않았다. HyTime같은 경우에는 스케줄링 메카니즘이 상세히 기술되어 있다고는 하지만 사용자의 인터랙션을 처리할 수 있는 메카니즘에 대한 기술이 부족하였다.

그럼으로 본 논문에서 제시한 DTD에서는 그런 단점들을 보완하여 사용자들이 DTD를 이해하고 사용하기가 편하도록 하였고, 기존의 어플리케이션들의 가장 큰 단점으로 지적되고 있는 멀티미디어 객체들의 스케줄링에 관한 사항들을 수정, 보완하여 사용자들이 의도하는 하이퍼미디어 문서를 표현하기에 부족함이 없도록 하였다. 그리고 멀티미디어 객체의 단순한 재생이 아닌 선택적인 구간 재생과 선택한 구간에 대한 재생 속도의 변경 또한 가능하도록 설계하였으며, 새로운 시간종속적 하이퍼미디어 모델을 구성함에 있어 이벤터의 개념을 도입함으로써 사용자의 인터랙션도 처리가 가능해졌다.

그러나 본 논문에서 제시하는 DTD를 이용하여 실제 문서를 작성하고 파서를 통한 검사를 한 후 동기화 메카니즘을 적용하여 분석을 실시했지만, 그것은 단지 SGML 구문에 관한 검증과 이론적인 면에서의 검증만 실시하였을 뿐 실질적인 프리젠테이션 시스템에서의 동작에 관한 확인은 하이퍼미디어 시스템이 구현 단계에 있어 일부만 실시한 상태이며 본 논문의 연구에서 제외하였다. 따라서 본 논문의 4장에서 제시한 시간종속적 하이퍼미디어 시스템이 빠른 시간내에 실질적인 시스템으로 구체화하여 브라우저상에서의 프리젠테이션과 검증을 실시한 후, 단점들을 보완할 수 있다면 더욱 효율적인 DTD가 될 수 있으리라 기대된다. 그리고 DTD와 DI의 구문 확인을 위해 파싱을 하는 과정에서 사용한 파서 ArcSGML에서 많은 문제점이 발견되었다. SGML을 이용한 어플리케이션의 설계에 있어서는 파서가 매우 중요한 역할을 담당하게 됨으로 그런 오류들을 빠른 시간내에 수정하고 보완해야만 할 것이다.

아울러 앞으로의 하이퍼미디어 시스템에서는 문서 자체에 프리젠테이션 도구에서의 메뉴와 같은 항목들이 포함될 수도 있을 것이다. 즉 문서 자체에서 사용자가 조건적인 사항들을 부여하면서 문서를 브라우징하는 형태가 될 것이다. 따라서 그러한 측면에서의 연구도 병행한다면 좀 더 완벽한 하이퍼미디어 시스템을 구현할 수 있을 것으로 기대된다.

참 고 문 헌

[1] 홍진석, HyTime을 이용한 하이퍼미디어 시스템의 개발, 연세대학교 석사 논문, 1994.

[2] Lynda Hardman,Dick C.A Bulterman, Guido Van Rossum, *The Amsterdam Hypermedia Model*, Commun. ACM 37, 2, pp.50-62, Feb. 1994.

[3] ISO 8879-1986 *(Standard Generalized Markup Language)*.

[4] Martin Bryan, *An Authors guide to the Standard Generalized Markup Language*, 1988.

[5] ISO/IEC DIS 10744 *(Information technology - Hypermedia / Time - based Structuring Language (HyTime)*. 1991.

[6] Frank Halasz, Mayer Schwartz, *The Dexter Hypertext Reference Model* SIGGRAPH 1990 (3), pp.95-133, Aug. 1990.

[7] C.M.Sperberg-McQueen and Lou Burand, *Guidelines For the Encoding and Interchange of Machine-Readable Texts*, 1990. ACH, ACL, ALLC.

[8] Steven J. Derose and David G.Durand, *Making Hypermedia Work*, Kluwer Academic publishers, 1994.

[9] John D. Koegel, Lloyd W. Rutledge, John L. Rutledge, Can Keskin *HyOctane: A HyTime Engine for an MMIS*. 1993 ACM Multimedia 93. Aug. 1993.

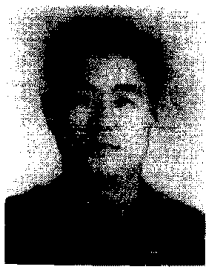
[10] IAN S. GRAHAM, *HTML SOURCEBOOK : A Complete Guide to HTML*, 1995.

[11] Y.K.Ko, Y.C.Choy, *Modeling for Interactive Presentation and Navigation of Time-Dependent Multimedia Information*, IEEE Workshop, pp. 143-149, Aug. 1995.

[12] John F. Koegel Buford, *Multimedia Systems*, ACM Press, 1994.

[13] Jessica Keyes(ed.), *Multimedia Handbook*, McGraw-Hill, 1994.

[14] Jakob Nielson, *Multimedia and Hypermedia*, AP Prof, 1995.



김 호 진

1982년 성균관대학교 성영학과 졸업(학사)

1985년 Florida 공대 대학원 전산학과 졸업(석사)

1997년 11월~현재 전술C4I개발단 기반체계처장

관심분야 : C4I, CBT 하이퍼미디어 시스템, SGML

이 수 진

직책 : 육군교육사령부 전산개발처 제도분석장교

1992년 육군사관학교 전산과 졸업(이학사)

1996년 연세대학교 대학원 컴퓨터과학과(이학석사)

1996년~현재 육군교육사령부 전산개발처 제도분석장교

관심분야 : 멀티미디어, 하이퍼미디어, SGML,

최 윤 철

1773년 서울대학교 전자공학과 졸업(공학사)

1975년 6월 공학 석사(Univ. of Pittsburgh)

1979년 6월 공학 박사(Univ. of Califonia, Berkeley, Dept. of IE&OR)

1979년 8월~1982년 7월 Lockheed사 및 Rockwell International사 책임 연구원

1982년 9월~1984년 1월 Univ. of Washington 전산학

1990년 9월~1992년 1월 Univ. of Massachusetts 연구교수

1984년~현재 연세대학교 컴퓨터과학과 교수

관심분야 : 멀티미디어, 하이퍼미디어, 지리정보 시스템(GIS)