SMIL, RDF, WIDL 문서의 통합 객체 모델링

(Integrated Object Modeling for SMIL, RDF, WIDL Documents)

김 상 은 † 하 얀 †† 김 용 성 †††
(Sang-Eun Kim) (Yan Ha) (Yong-Sung Kim)

요 약 XML은 다양하게 응용할 수가 있어 여러 분야에서 널리 사용되고 있다. 그러나 이러한 응용들에 대해서 통합적으로 관리해 주는 시스템은 제안된 바 없어, 각각의 응용에 대해 별개의 언어로 사용되고 있다. 따라서, 본 논문은 XML의 다양한 응용 중에 웹을 기반으로 하는 대표적인 응용인 SMIL, RDF, WIDL에 대해, 이들의 DTD와 문서 인스턴스를 통합하는 객체 모델링을 하고자 한다. 각 XML 응용에 대해 객체 모델링 규칙과 알고리즘을 통합할 수 있는 시스템을 제안한다. 이를 통해 XML 종류에 상관없이 웹 기반 XML 응용의 구조를 쉽게 파악할 수 있으므로 문서 생성을 용이하게 하며, 객체지향 스키마를 쉽게 생성할 수 있으므로 객체지향 데이터베이스 문서관리의 기반이 될 것이다.

Abstract For having various applications, XML is widely used in various fields. But, there have not been proposed a system to integrate and manage XML and its applications together. In this paper, we propose methods to integrate web-based XML applications(SMIL, RDF, WIDL) and for object-oriented modeling of each DTD and document instance. We propose an system to merge object modeling of XML applications. With the proposed integrating algorithm, we can not only easily analysis various web-based XML applications which is represented by complex tags but also generate object-oriented schema for each document and store it to OODBMS.

1. 서 론

XML(eXtensible Markup Language)은 특정 응용 목적에 맞게 엘리먼트, 애트리뷰트, 엔티티를 정의할 수 있는 융통성(flexibility)을 갖고 있으며, 대표적인 예로 SMIL(Synchronized Multimedia Integration Language), RDF(Resource Description Framework), WIDL (Web Interface Definition Language) 등이 있다[1]. 이들 XML 응용들은 웹을 기반으로 한 형태이기 때문에 앞으로 그 사용이 점차 급증하리라 예상된다. 이들 응용에 대해 간략하게 살펴보면, SMIL은 그래픽, 오디오 등 멀티미디어 내용을 웹에 전송할 수 있도록 지원해 주는 표준화된 문법이며[2], RDF는 메타데이타를 처리하기 위한 기반으로 웹에서 기계가 읽을 수 있는 정보를 교환하는 응용들(applications) 간에 상호운영성(interoperability)을 제공하는 규정이다[3]. 그리고, WIDL은 웹 데이타와 서비스를 위한 응용 프로그램 인터페이스(APIs)를 정의하기 위한 메타데이타 문법이다. 지금까지 업무 응용 분야에서 웹 데이타에 직접 접근하는 문제에 대해서는 거의 무시되어 왔으나, 최근에는 WIDL로 인하여 웹 자원을 원격 시스템에서 표준 웹 프로토콜에 접근할 수 있는 인터페이스로 기술되어 사용되고 있다. WIDL의 목적은 표준 웹 프로토콜에 대한 요청/응답 상호작용을 표현하는 일반적인 방법을 제공하고, 웹에서 다양한 통합 플랫폼으로 이용될 수 있는 HTML/XML 문서와 형태를 가진 지원들을 상호작용할 수 있게 자동화하는데 있다. WIDL에서 정의한 서비스는 웹의 자원이 수정 없이 다른 업무 시스템과 잘 통합되도록 웹 내용을 프로그램 변수에 사상되어야 한다[4, 5, 6].

이러한 특징을 갖고 있는 WIDL 역시 객체지향적인

† 종신회원 : 대경대학 컴퓨터통신계열 교수
sekim@tk.ac.kr
†† 정 회 원 : 중앙대학교 정보통신연구소 연구전담교수
hayan@object.cse.cau.ac.kr
††† 종신회원 : 전북대학교 컴퓨터과학과 교수
yskim@moak.chonbuk.ac.kr
논문접수 : 2000년 5월 2일
심사완료 : 2000년 10월 9일

XML 응용이기는 하나 이에 대한 객체 모델링과 스키마 형태에 대해 제안된 바가 없다.

따라서, 본 논문에서는 대표적인 웹 기반 XML 응용인 SMIL, RDF, WIDL을 통합하여 객체모델링하기 위한 알고리즘을 제안한다. 그리고, XML 응용에 따른 구별 없이 DTD에 공통적으로 적용되는 규칙을 제안한다.

논문 구성은 다음과 같다. 2장은 관련 연구를 설명하고, 3장에서는 UML의 각종 다이어그램과 웹 기반 XML 응용의 엘리먼트 태그들에 대해 설명하며, 4장에서는 SMIL, RDF, WIDL 문서를 통합한 객체 모델링을 제안한다. 이를 위해 모델링 규칙, 모델링 함수, 모델링 알고리즘을 제안한다. 그리고, 5장에서는 각 XML 응용과 DTD를 통합한 시스템을 제안하며, 적용 결과를 제시한다. 끝으로 6장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

XML 문서를 위한 표준화된 API를 제공하여 웹 문서에 접근하고 조작하기 위한 방법으로 DOM(Document Object Model)[1]이 있는데, 이것은 문서의 논리적 구조를 정의하는 객체 기반 구조이기는 하나 트리 계층 구조의 형태이므로 클래스의 세부적인 정보인 애트리뷰트나 집단화 관계 등을 제대로 표현하지 못한다.

이 외에 DTD에 대한 객체 모델링 연구는 XOMT와 UML 클래스 다이어그램이 있다. XOMT는 OMT의 기능을 많이 확장하였으나, 애트리뷰트를 정의하는 클래스 등 객체지향 개념에 잘 맞지 않는 부분이 있다. 이에 비해 UML 클래스 다이어그램은 DTD를 객체지향 개념에 어긋나지 않게 객체 모델링을 하였다[7].

따라서, 본 연구는 [7]의 SGML DTD를 UML 클래스 다이어그램에 사상시키는 규칙을 기반으로 XML DTD에 맞게 사상시키는 알고리즘을 제안하며, 이를 통해 UML 클래스 다이어그램을 생성한다.

한편, XML 문서에 대한 모델링으로서 [8]이 제안되었으며, 이를 확장하여 SMIL 문서에 대해 UML 사용사례, 순서 및 협력 다이어그램을 이용하여 동기화 모델링[9]과 RDF 자원에 따른 시멘틱 분석으로 UML 클래스 다이어그램으로 객체 모델링이 제안[10]된 바 있었다.

그러나, 아직까지 SMIL, RDF 이외의 XML 응용에 대한 구체적인 모델링 방법이 제시된 바 없고, 이들 웹 기반 응용의 통합에 관해서 연구된 바가 없다. 따라서, 본 연구에서는 웹 기반 XML 응용을 통합하고, XML DTD를 객체 모델링하며, 각 XML 응용 문서에 대해 객체 모델링하고 객체지향 코드를 생성할 수 있도록 하였다. 이미 연구가 진행되어 온 SMIL, RDF 문서에 WIDL 문서를 추가하여 웹기반 XML 응용을 위한 통합 시스템을 제안한다.

3. UML과 SMIL, RDF, WIDL

본 장에서는 UML의 대표적인 다이어그램과 웹 기반 XML 응용에 대해 살펴본다.

3.1 UML의 각종 다이어그램

UML에는 사용사례 다이어그램, 순서와 협력 다이어그램, 클래스 다이어그램, 그리고 컴포넌트 다이어그램 등이 있다.

3.1.1 사용 사례 다이어그램

사용 사례 다이어그램은 시스템의 외부 행위자와 시스템이 제공하는 기능인 사용 사례(Use case)와의 관련성을 나타낸다. 하나의 사용 사례는 그 시스템이 제공하는 기능 중의 하나를 나타내며, 클래스 다이어그램 등을 삽입(insert)할 수 있다[11].

다음은 사용 사례 다이어그램의 구성요소이다.

(1) 액터(Actor)

시스템과 서브시스템, 혹은 클래스와 상호작용을 하는 외부 사람, 프로세스 등을 말한다.

(2) 사용 사례(Use Case)

시스템의 내부 구조를 나타내지 않는 하나의 연관성 있는 행동의 단위를 말한다.

3.1.2 순서 다이어그램

순서 다이어그램은 다른 객체들 사이의 메시지 흐름의 순서에 초점을 두는 것으로, 표기는 2개의 축을 중심으로 한다. 수직 축은 시간의 흐름을 나타내고 수평 축은 해당 객체를 나타낸다.

다음은 순서 다이어그램에서 표현되는 요소이다.

(1) 객체(Object)

직사각형으로 나타내고, 이름부분은 밑줄을 치고, 생명선(lifeline)은 점선으로 된 수직선으로 표현한다.

(2) 사건 식별자(Event identifier)

메시지가 일어나는 사건을 참조할 때 사용한다.

(3) 시간 제한(Time constraint)

사건들(events) 사이에 관련 있는 시간을 나타낼 때 사용한다.

(4) 메세지(Message)

객체들 간의 상호작용을 나타내는 것으로 객체의 생명선 사이의 수평선이 된다. 메시지의 종류는 크게 2가지인데, Peer-to-Peer 메시지는 하나의 객체에 보내는 메시지이고, Broadcast 메시지는 하나 이상의 다른 객체에게 동시에 보내는 경우이다.

3.1.3 협력 다이어그램

순서 다이어그램과 같은 정보를 갖는 것으로 협력 다이어그램이 있다. 그러나, 협력 다이어그램은 메시지의 순서보다는 객체들 간의 정적 구조에 초점을 맞추고 있으며, 객체들간의 관계를 나타내는 메시지는 일련번호를 갖는다.

3.1.4 클래스 다이어그램

XML 응용들을 모델링 하기 위해 공통적으로 사용하는 UML 클래스 다이어그램의 구성 요소로는 클래스, 애트리뷰트, 오퍼레이션, 컴포지션 관계와 'or' 제한조건이 있다[12]. 이에 대한 설명은 다음과 같다.

(1) 클래스(Class)

클래스는 클래스 이름, 애트리뷰트 리스트(Attribute list)와 오퍼레이션 리스트(Operation list) 부분으로 구성된다. 애트리뷰트 리스트는 클래스의 멤버들을 나타내고, 오퍼레이션 리스트는 멤버들을 다루는 함수를 나타낸다.

(2) 컴포지션(Composition)과 'or' 관계

집단화(Aggregation) 관계는 수퍼클래스와 서브클래스가 'part-of' 관계이며 '◇'로 표기한다. 강한 집단화 관계로 컴포지션이 있는데, 이것은 서브클래스가 수퍼클래스와 같은 생명주기를 갖는 강한 형태의 구성 관계를 나타낸다. 표기는 '◆'로 한다.

2개 이상의 집단화 관계에서 선택적으로 서브클래스가 발생하는 경우 'or' 관계를 갖는다. 이 때, 여러 개의 집단화 관계를 점선으로 연결하고 '{or}'라는 제한조건을 준다.

3.1.5 컴포넌트 다이어그램

물리적인 관점에 속하는 다이어그램의 하나로, 소프트웨어 컴포넌트들 간의 의존관계를 보여준다.

컴포넌트 다이어그램의 구성요소는 다음과 같다.

(1) 컴포넌트(Component)

시스템의 물리적이고 대체 가능한 부분으로 인터페이스를 준수하고 구현한다. 한쪽 측면에 작은 사각형 2개를 가진 직사각형으로 표현된다.

(2) 인터페이스(Interface)

컴포넌트의 서비스를 명세화하는 오퍼레이션들을 모아놓은 것으로 작은 원으로 표기하며, 컴포넌트와 굵은 선으로 연결된다.

(3) 의존 관계(Dependency)

한 쪽 컴포넌트의 변화가 다른 컴포넌트에 영향을 줄 수 있는 관계로 점선으로 나타내되 방향성을 나타낼 수 있다.

3.2 웹 기반 XML 응용

대표적인 웹 기반 XML 응용으로는 SMIL, RDF, WIDL이 있다. 본 절에서는 이들 문서에서 사용되는 엘리먼트 태그들을 기술한다.

3.2.1 SMIL

SMIL은 크게 <head>와 <body>부분으로 구성된다. <head>부분은 순서에 따라 제시하는 것과 무관한 정보이며, <body>부분은 문서의 시간상 연결 행동과 관련된 정보를 포함한다.

이에 대한 태그의 종류와 이름은 표 1과 같다.

표 1 SMIL의 태그들

구분	태그종류	태그이름
<head>부분	영역태그	<layout>
		<region>
	선택태그	<switch>
<body>부분	미디어태그	<ref>
		<animation>
		<audio>
		<image>
		<video>
		<text>
		<textstream>
	동기태그	<par>
		<seq>
	하이퍼링크	<a>
		<anchor>
	선택태그	<switch>

3.2.2 RDF

RDF 표현에 의해 기술되는 것을 자원이라 하며, 이에 해당하는 요소로는 클래스, 프로퍼티, 제한조건 프로퍼티가 있다. 프로퍼티는 애트리뷰트나 관계성을 나타내며, 제한조건 프로퍼티는 그것의 인스턴스가 제한조건을 포함한 RDF 스키마 구조나 프로퍼티를 말한다.

RDF 자원의 태그들은 표 2와 같다.

표 2 RDF 자원의 태그들

구분	태그종류	태그이름
클래스	클래스태그	<rdf:Description>
프로퍼티	타입태그	<rdf:type>
	서브클래스태그	<rdfSubclassOf>
	설명태그	<rdf:label>
		<rdf:comment>
제한조건 프로퍼티	제한조건태그	<rdfs:range>
		<rdfs:domain>
<rdfs:Container>	순서태그	<rdfs:Bag>
	비순서태그	<rdf:Seq>
	선택태그	<rdf:Alt>

3.2.3 WIDL

태그의 중요한 정도에 따라 주요 태그와 그렇지 않은 태그로 구별한다. 이에 대한 태그 종류와 이름은 표 3과 같으며, 세부적인 내용은 다음과 같다.

표 3 WIDL의 태그들

구분	태그종류	태그이름
주요 (major)	인터페이스태그	<WIDL>
	서비스태그	<SERVICE>
	매개변수태그	<BINDING>
기타 (minor)	바인딩매개변수태그	<VARIABLE>
	에러메시지태그	<CONDITION>
	영역태그	<REGION>

(1) <WIDL>

<WIDL>은 부모 엘리먼트로서, 인터페이스를 정의한다. 여기서 인터페이스란 관련된 서비스와 바인딩의 그룹이다. <WIDL> 엘리먼트에 해당하는 애트리뷰트는 NAME, VERSION, TEMPLATE, BASEURL, OBJMODEL 이다.

(2) <SERVICE>

웹 서비스를 정의하는 것으로 입력 매개변수의 집합을 갖고 프로세싱을 수행하며, 동적으로 HTML, XML 이나 텍스트 문서를 반환한다. <SERVICE> 엘리먼트의 애트리뷰트는 추상적인 서비스 이름을 서비스의 실제 URL(Uniform Resource Locator)에 사상시키고, 서비스에 접근하기 위해 사용한 HTTP 방법을 규정하며, 입력 매개변수와 출력 매개변수를 위한 바인딩을 나타낸다.

(3) <BINDING>

서비스를 위한 입력(Input)과 출력(Output) 매개변수(parameter)를 정의한다. 입력 바인딩은 웹 자원으로 제공된 데이타를 기술한다. 이것은 HTML 형태에서 입력 필드와 유사하다. 출력 바인딩은 주어진 입력 변수를 갖고 웹 자원에 접근한 결과로서 반환되는 출력 문서로부터 사상되는 데이타 엘리먼트를 기술한다. 많은 경우에 출력 바인딩은 출력 문서의 엘리먼트들의 부분 집합에만 사상된다.

(4) <VARIABLE>

입력과 출력 바인딩 안의 매개변수를 정의한다. <VARIABLE> 엘리먼트에 해당하는 애트리뷰트는 NAME, VALUE, USAGE, TYPE, FORMNAME, OPTIONS 등이 있다.

(5) <CONDITION>

호출 프로그램에 반환되는 데이타를 얻기 위해 성공과 실패를 규정하는 출력 바인딩에서 사용된다. 서비스가 실패했을 때는 호출 프로그램에 반환되는 에러 메시지를 정의한다.

(6) <REGION>

문서 안의 영역을 정의하기 위해 출력 바인딩에서 사용된다. 이것은 한 페이지의 엘리먼트들 사이에 위치할 수 있는 구조 안의 정보의 가변 배열을 반환하는 서비스에서 유용하다.

4. 웹기반 XML 문서 모델링

본 장에서는 대표적인 웹 기반 XML 문서들에 대해 UML 클래스 다이어그램을 생성하기 위한 모델링 규칙, 각 클래스의 멤버함수, 그리고, 모델링 알고리즘을 제안한다.

4.1 모델링 규칙

다음은 웹 기반 XML 응용 문서 인스턴스를 UML 클래스 다이어그램의 집합으로 사상시키는데 필요한 규칙들이다.

(1) 엘리먼트

시작 태그가 되는 엘리먼트에 대한 규칙들을 정의하면 다음과 같다.

[규칙 1] SMIL의 미디어 태그, 하이퍼링크 태그, 선택 태그, RDF의 <RDF:Description> 태그, WIDL의 시작 태그가 되는 엘리먼트는 클래스가 된다.

이 때, 같은 타입의 클래스가 반복 생성되는 경우 각 클래스 이름에 순차 번호를 붙여 이를 구별해 준다.

(2) 애트리뷰트와 값

애트리뷰트와 그 값에 관한 규칙은 다음과 같다.

[규칙 2] RDF 스키마의 애트리뷰트, WIDL의 엘리먼트 태그 내에 애트리뷰트와 값은 해당 클래스의 Private 애트리뷰트와 값이 된다.

(3) 엘리먼트들 간의 관계

엘리먼트와 그 안에 포함된 엘리먼트들 간의 관계는 다음과 같이 정의한다.

[규칙 3] RDF의 <rdfs:Container>, WIDL의 엘리먼트 태그 안에 오는 엘리먼트는 클래스와 하위 클래스들 간의 컴포지션 관계가 된다.

[규칙 4] RDF의 <rdfs:subClassOf>는 엘리먼트 클래스와 하위 클래스들 간의 일반화 관계가 된다[9].

4.2 모델링 함수

SMIL, RDF, WIDL 문서를 UML 클래스 다이어그램으로 사상시켰을 때, 각 클래스의 오퍼레이션 리스트

부분에 삽입되는 멤버 함수들을 각각 다음과 같다. 이들 멤버함수들은 해당 클래스 다이어그램의 구조 파악과 변형을 용이하게 해준다.

4.2.1 SMIL의 멤버 함수

SMIL 문서의 경우 순서 다이어그램의 객체로부터 클래스를 추출하므로 클래스의 발생 순서에 관한 동기화 함수가 필요하며, 참조 클래스에 관한 함수도 필요하다. 이에 대한 내용은 표 4와 같다[9].

표 4 SMIL 클래스의 멤버함수

함수명	설명
par_o()	동시 발생하는 클래스들
seq_o()	다음에 발생하는 클래스
href_o()	해당클래스 수행동안 참조될 클래스
anchor_n()	해당클래스 수행동안 참조될 클래스 수

4.2.2 RDF의 멤버 함수

다음은 RDF 자원의 클래스로 변환시켰을 때, 이에 해당하는 멤버함수이다[10]. 기본적으로 수퍼 클래스와 서브클래스를 나타내는 함수와 해당 클래스에 포함되는 애트리뷰트를 나타내는 함수가 필요하다. 그리고, 수퍼 클래스를 구성하는 하위클래스들 간의 관계에 관련된 함수가 필요하다.

표 5 RDF 자원 클래스의 멤버함수

함수명	설명
p()	수퍼클래스
c()	서브클래스
cons()	연결자의 종류
r()	서브클래스 순서
m()	애트리뷰트

4.2.3 WIDL의 멤버 함수

WIDL의 경우 기본적으로 클래스와 애트리뷰트와 관련된 함수 이외에 서비스 개수와 바인딩의 입출력 매개변수를 정의하는 함수가 필요하다. WIDL에 의해 생성된 클래스의 멤버 함수는 표 6과 같다

표 6 WIDL 클래스의 멤버함수

함수명	설명
p()	수퍼클래스
c()	서브클래스
s()	서비스 개수
i()	바인딩의 입력매개변수
o()	바인딩의 출력매개변수
m()	애트리뷰트

4.3 모델링 알고리즘

본 절에서는 RDF나 WIDL 같은 XML 문서 인스턴스를 입력하여 UML 클래스 다이어그램을 생성하는 알고리즘을 제안한다. 단, SMIL의 경우에는 순서 다이어그램으로부터 객체를 추출하여 클래스로 변환하여 클래스 다이어그램을 생성하는 것으로 이미 [10]에서 언급하였다.

create_DI_ClassDiagram()

입력: XML 문서
출력: UML 클래스 다이어그램

```
begin
{
루트 클래스 생성
// WIDL, Resource 중 하나 생성
for ( 애트리뷰트 리스트 개수)
    애트리뷰트와 값 삽입
멤버함수 삽입

for ( 시작 태그나 <rdf:Description> 개수 )
 {
    클래스 생성
    for ( 애트리뷰트 리스트 개수)
          애트리뷰트와 값 삽입
    멤버함수 삽입
    if (루트클래스 == WIDL)
      컴포지션 관계 형성
    else
      일반화 관계 형성
      // RDF 자원은 Resource로부터  상속
      while not (</rdf:Description>)
      { // RDF 자원에 대해
        if (<rdf:type>)
            클래스 타입 정의
        else if (<rdf:subClassOf>)
              일반화 관계 형성
        else if (<rdf:Container>)
              컴포지션 관계 형성
        else // WIDL 문서에  대해서
            {
             if ( 종료 태그 표시 (/)가 없으면)
                for (시작 태그 개수)
                 {
                  클래스 생성
                  for (애트리뷰트 리스트 개수)
                      애트리뷰트와 값 삽입
                  멤버함수 삽입
                  컴포지션 관계 형성
                  }
             }
        }
 }
}
end;
```

5. 통합 시스템

본 장에서는 4장에서 제안한 각 XML 응용에 대한 모델링을 통합한 알고리즘과 시스템을 제안한다. 이에 앞서, 각 응용 XML DTD로부터 클래스 다이어그램을 생성하기 위해 공통적으로 적용되는 모델링 규칙과 알고리즘을 제안한다.

5.1 XML DTD 클래스 다이어그램

일반적인 XML DTD에서 링크 부분을 제외한 문법을 이용하여 XML의 응용 DTD에 대해 클래스 다이어그램을 사상시키는 모델링 규칙을 제안한다. XML DTD는 크게 엘리먼트와 애트리뷰트 리스트 선언 부분으로 구성된다. 엘리먼트 선언은 선언 부분과 내용 부분으로 구성되며, 내용 부분은 '( )'와 발생 지시자, 엘리먼트들 간의 ',' 또는 '|' 관계로 구성되어 있다. 이러한 XML DTD는 클래스 다이어그램의 클래스, 애트리뷰트, 다중성, 컴포지션 관계, 그리고 'or' 제한조건으로 사상된다.

XML DTD의 각 요소에 대한 클래스 다이어그램으로 사상되는 사상 테이블은 다음과 같다.

표 9 DTD로부터 클래스다이어그램으로의 사상 테이블

<table>
<tr><th colspan="4">XML DTD</th><th colspan="2">UML 클래스 다이어그램</th></tr>
<tr><td rowspan="4">엘리먼트 선언</td><td>선언 부분</td><td colspan="2">엘리먼트</td><td colspan="2">클래스</td></tr>
<tr><td rowspan="3">내용 부분</td><td>연결자</td><td>'|'</td><td>컴포지션 관계</td><td>'or' 제한조건</td></tr>
<tr><td colspan="2">발생지시자</td><td colspan="2">다중성</td></tr>
<tr><td colspan="2">'( )'</td><td colspan="2">임시 클래스</td></tr>
<tr><td>애트리뷰트 리스트선언</td><td colspan="3">애트리뷰트</td><td>Private 애트리뷰트</td></tr>
</table>

위의 사상 테이블에 의해 각종 XML 응용 DTD를 입력하여 UML 클래스 다이어그램을 생성하는 알고리즘은 다음과 같다.

create_DTD_ClassDiagram()

입력: XML DTD
출력: UML 클래스 다이어그램

```
begin
{
  while ( DTD 요소 )
  {
  if ( 선언된 엘리먼트 )
    {
      클래스 생성
      애트리뷰트와 값 삽입
      if ( 내용 부분에 '( )' )
        {
          클래스(brace) 생성
          컴포지션 관계 형성
          발생 지시자 처리
          for ( 엘리먼트 개수 )
              {
              클래스 생성
              애트리뷰트와 값 삽입
              컴포지션 관계 형성
              }
      제한조건(or) 부여
    }
   }
  }
}
end
```

5.2 통합 알고리즘

웹 기반 XML 응용을 통합하는 알고리즘은 다음과 같다. 본 알고리즘은 먼저, DTD와 문서 인스턴스를 구별하고, 각 문서 인스턴스에 대해 문서의 타입에 따라 별도의 다이어그램을 생성하도록 한다. 예를 들어, SMIL의 경우에는 사용사례, 순서, 협력 다이어그램이 추가되어 이를 통해 클래스 다이어그램이 생성되도록 한다.

입력: XML DTD, SMIL 문서, RDF 문서, WIDL 문서
출력: 객체지향 코드

```
begin
{
  parse_XML(DTD 또는 문서 인스턴스)
 // XML DTD나 문서를 파싱한다.
  if ( DTD )
     create_DTD_ClassDiagram()
      // 5.1 모델링 알고리즘을 적용하여
      // DTD 클래스 다이어그램을 생성한다.
  else
  { // 문서 인스턴스
    if (문서 타입 == SMIL)
    // SMIL 문서인 경우
    then
        {
         create_UsecaseDiagram()
         // 각 영역을 사용사례로 구별한다.
         for ( Usecase 수 )
           { //  각 사용사례에 대해
             insert_SequenceDiagram()
              // 순서 다이어그램을 삽입한다.
             convert_CollaborationDiagram()
              // 순서 다이어그램으로부터 협력
              // 다이어그램으로 변환시킨다.
```

```
                convert_ClassDiagram()
                // 순서 다이어그램에서 클래스를 추출
                // 하여 클래스 다이어그램을 생성한다.
                 }
        }
    else
        create_DI_ClassDiagram()
            // 4.3 모델링 알고리즘을 적용하여
            // XML 문서 클래스 다이어그램을 생성한다.
  }

  while not (ClassDiagram)
  { // 모든 클래스다이어그램에 대해
     while not (Class)
          { // 모든 클래스에 대해
              deploy_ComponentDiagram()
            // 컴포넌트 다이어그램을 할당시킨다.
              generate_Code()
              // 객체지향 코드를 생성한다.
          }
  }
}
end;
```

5.3 시스템 구성

XML 응용에 대한 DTD와 문서 인스턴스가 입력되었을 때, 다이어그램과 객체지향 코드 생성에 관한 시스템 구성도는 그림 1과 같다.

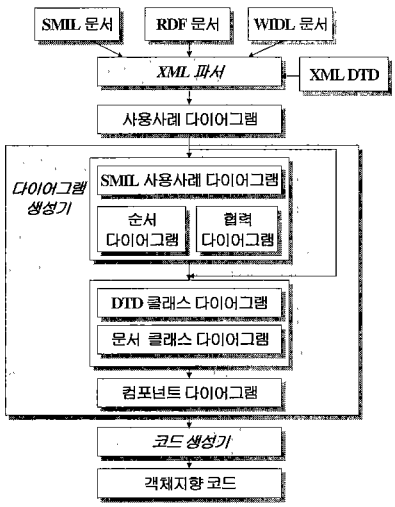

그림 1 XML 응용 통합 시스템 구성도

(1) XML 파서

XML DTD와 문서 인스턴스를 입력받아 문법상의 오류를 검사하고 파싱된 테이블을 생성한다.

(2) 사용사례 다이어그램

XML의 다양한 응용은 UML의 각 사용사례로 구별하여 처리한다. 이에 대한 것은 그림 2와 같다.

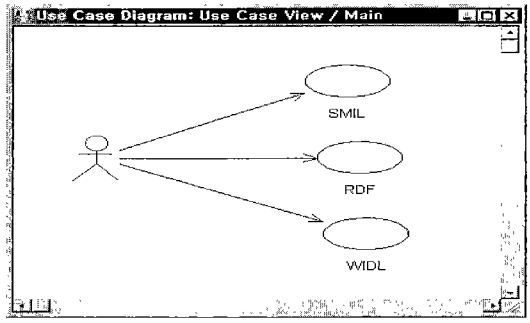

그림 2 사용사례 다이어그램

(3) 다이어그램 생성기

XML 인스턴스에 사용된 태그가 SMIL DTD에서 정의한 태그인 경우 사용 사례와 순서 다이어그램을 먼저 생성하고 이로부터 협력 다이어그램과 클래스 다이어그램을 자동 생성한다. DTD와 RDF, WIDL 문서에 대해서는 별도의 다이어그램 생성 없이 바로 클래스 다이어그램을 생성한다.

(4) 코드 생성기

XML DTD와 문서 인스턴스로부터 생성된 각각의 클래스 다이어그램으로부터 컴포넌트 다이어그램을 생성하고 각 컴포넌트에 대해 객체지향 코드인 C++코드를 생성한다.

5.4 적용 결과

본 알고리즘에 의해 DTD와 문서 인스턴스를 입력으로 하여 그 적용 결과를 살펴본다.

5.4.1 DTD 클래스 다이어그램

본 연구에서는 XML DTD를 입력시켜 클래스 다이어그램을 생성한다.

(1) XML DTD

DTD의 한 예로 WIDL를 사용하였으며, WIDL DTD는 다음과 같다.

```
<!ELEMENT WIDL (SERVICE | BINDING )*>
<!ATTLIST WIDL
        NAME     CDATA    #IMPLIED
        VERSION (1.0 | 2.0 | … ) "2.0"
        TEMPLATE CDATA #IMPLIED
        BASEURL CDATA #IMPLIED
        OBJMODEL ( wmdom | … ) "wmdom"
```

```
>
<!ELEMENT SERVICE EMPTY>
<!ATTLIST SERVICE
        NAME CDATA #REQUIRED
        URL CDATA #REQUIRED
        METHOD (Get | Post) "Get"
        INPUT CDATA #IMPLIED
        OUTPUT CDATA #IMPLIED
        AUTHUSER CDATA #IMPLIED
        AUTHPASS CDATA #IMPLIED
        TIMEOUT CDATA #IMPLIED
        RETRIES CDATA #IMPLIED
>
<!ELEMENT BINDING (VARIABLE | CONDITION |
        REGION)* >
<!ATTLIST BINDING
        NAME  CDATA #REQUIRED
        TYPE (INPUT|OUTPUT) "OUPUT"
>
<!ELEMENT VARIABLE EMPTY>
<!ATTLIST VARIABLE
        NAME CDATA #REQUIRED
        FORNAME CDATA #REQUIRED
        TYPE  (String | String[] | String[][])
                "String"
        USAGE  (Default | Header | Internal)
                "Function"
        REFERENCE CDATA #IMPLIED
        VALUE CDATA #IMPLIED
        MASK CDATA #IMPLIED
        NULLOK          #BOOLEAN
>

<!ELEMENT CONDITION EMPTY>
<!ATTLIST CONDITION
        TYPE (Success | Failure | Retry) "Success"
        REF CDATA #REQUIRED
        MATCH CDATA #REQUIRED
        REBIND CDATA #IMPLIED
        SERVICE CDATA #IMPLIED
        REASONREF CDATA #IMPLIED
        REASONTEXT CDATA #IMPLIED
        WAIT CDATA #IMPLIED
        RETRIES CDATA #IMPLIED
>
<!ELEMENT REGION EMPTY>
<!ATTLIST REGION
        NAME CDATA #REQUIRED
        START CDATA #REQUIRED
        END CDATA #REQUIRED
>
```

(2) DTD 클래스 다이어그램

(1)의 DTD를 UML 클래스 다이어그램으로 변환시킨 결과는 그림 3과 같다.

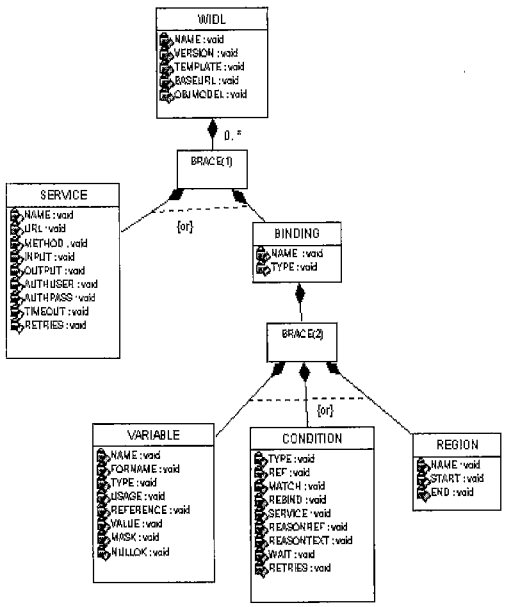

그림 3 DTD 클래스 다이어그램

5.4.2 문서 클래스 다이어그램

웹 기반 XML 응용에 해당하는 문서 인스턴스를 입력으로 하였을 경우 클래스 다이어그램과 객체지향 코드를 생성한다.

(1) XML 문서 인스턴스

소포 추적 서비스(Package Tracking Service)에 대한 WIDL 표현은 다음과 같다.

```
<WIDL NAME="FedExShipping"
        TEMPLATE="Shipping"
        BASEURL="http://www.FedEx.com"
        VERSION="2.0">

<SERVICE NAME="TrackPackage" METHOD="GET"
            URL="/cgi-bin/tract_it"
            INPUT="TrackInput"
            OUTPUT="TrackOutput" />

<BINDING NAME="TrackInput" TYPE="INPUT">
    <VARIABLE NAME="TrackingNum"
        TYPE="String" FORMNAME="trk_num" />
    <VARIABLE NAME="DestCountry"
        TYPE="String"
        FORMNAME="dest_cntry" />
    <VARIABLE NAME="ShipDate" TYPE="String"
```

```
        FORNAME="ship_date" />
</BINDING>
<BINDING NAME="TrackOutput" TYPE="OUTPUT">
    <CONDITION TYPE="FAILURE"
      REFERENCE="doc.title[0].text"
      MATCH="FedEx Warning Form"
      REASONREF="doc.p[0].text['&']" />
    <CONDITION TYPE="SUCCESS"
      REFERENCE="doc.title[0].text"
      MATCH="FedEx Airbill:"
      REASONREF="doc.p[1].value" />
    <VARIABLE NAME="disposition"
      TYPE="String"
      REFERENCE="doc.h[3].value" MASK="$" />
    <VARIABLE NAME="deliveredOn"
      TYPE="String"REFERENCE="doc.h[5].value"
      MASK="%%%$" />
    <VARIABLE NAME="deliveredTo" TYPE="String"
      REFERENCE="doc.h[7].value" MASK="*:" />
</BINDING>
</WIDL>
```

(2) 문서 인스턴스 클래스 다이어그램

(1)을 입력으로 하여 UML 클래스 다이어그램을 생성하면 그림 4와 같다.

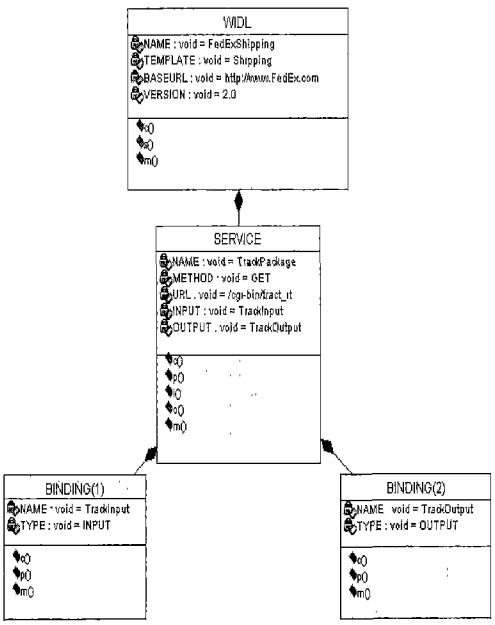

그림 4 문서 인스턴스 클래스 다이어그램

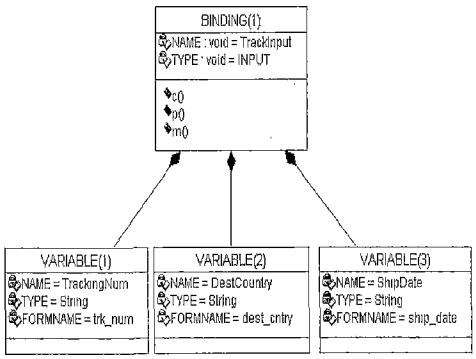

그림 5 "BINDING(1)"의 클래스 다이어그램

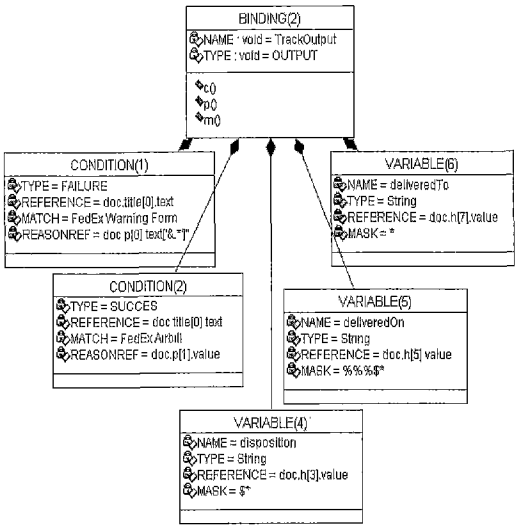

그림 6 "BINDING(2)"의 클래스 다이어그램

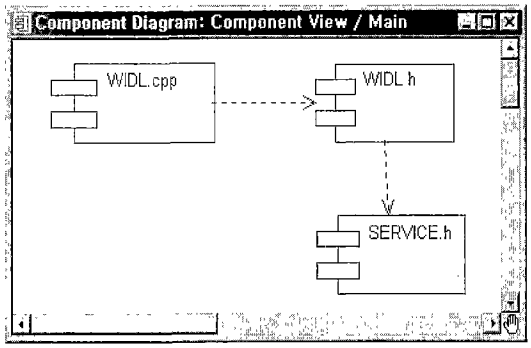

그림 7 "WIDL" 클래스의 컴포넌트 다이어그램

그림 4에서 클래스 "BINDING(1)"과 "BINDING(2)"에 대한 구체적인 그림은 각각 그림 5, 그림 6과 같다.

(3) 문서 인스턴스 객체지향 코드

그림 4의 클래스 "WIDL"에 대한 컴포넌트 다이어그램은 그림 7과 같으며, 본 논문에서는 C++ 코드를 사용하였다. 파일 "WIDL.cpp"는 파일 "WIDL.h"에 대해 컴파일 의존성을 갖고, "WIDL.h"는 "SERVICE.h"에 대해 의존성을 갖는다.

부록에는 "WIDL" 클래스에 대한 코드 "WIDL.cpp"와 "WIDL.h"를 제시한다.

6. 결론 및 향후 연구과제

본 논문은 XML의 웹 기반 응용인 SMIL, RDF, WIDL을 통합하여 객체 모델링하기 위한 알고리즘과 시스템을 제안하였다. 이를 위해 XML 각 응용으로부터 UML 클래스 다이어그램에 사상시키는 규칙과 각 클래스의 멤버 함수를 정의하였다.

그러므로, 본 연구에 의해 XML 문서의 구조 파악이 용이하여 문서 개발자는 손쉽게 다양한 응용의 XML 문서 인스턴스를 생성할 수 있고, 웹 기반 XML DTD나 문서 인스턴스를 입력하여 객체 모델링과 코드를 얻게되므로 데이타베이스 설계자는 XML 응용에 따른 태그의 종류나 문법을 해석할 필요 없이 스키마를 생성할 수가 있다. 본 연구의 가장 큰 의의는 객체지향 스키마를 쉽게 생성할 수 있으므로 웹 기반 XML의 객체지향적인 문서 관리에 기반이 된다는 것이다.

향후 연구 과제로는 웹 기반 XML 응용 뿐 만 아니라 다른 응용을 모두 통합한 XML DTD 다이어그램과 문서 인스턴스 다이어그램을 생성하고, 이를 기반으로 각종 OODBMS 종류에 따른 스키마를 자동 생성하는 것이다.

부 록

WIDL.cpp

```
// WIDL
#include "WIDL.h"

// Class WIDL
WIDL::WIDL()

WIDL::WIDL(const WIDL &right)

WIDL::~WIDL()

WIDL & WIDL::operator=(const WIDL &right)

int WIDL::operator==(const WIDL &right) const

int WIDL::operator!=(const WIDL &right) const

//## Other Operations(implementation)
void WIDL::c ()
void WIDL::s ()
void WIDL::m ()
```

WIDL.h

```
#ifndef WIDL_h
#define WIDL_h 1

// SERVICE
#include "SERVICE.h"

class WIDL
{
  public:
    //## Constructors (generated)
      WIDL();
      WIDL(const WIDL &right);

    //## Destructor (generated)
      ~WIDL();

    //## Assignment Operation (generated)
      WIDL & operator=(const WIDL &right);
    //## Equality Operations (generated)
      int operator==(const WIDL &right) const;
      int operator!=(const WIDL &right) const;

    //## Other Operations (specified)
      void c ();
      void s ();
      void m ();

  //## Get and Set Operations for Associations (generated)
      const SERVICE * get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE * value);

      const SERVICE get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE value);

      const SERVICE get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE value);

      const SERVICE get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE value);

      const SERVICE get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE value);

      const SERVICE get_the_SERVICE () const;
      void set_the_SERVICE (SERVICE value);
```

```
  private:
 //## Get and Set Operations for Class Attributes (generated)
      const void get_NAME () const;
      void set_NAME (void value);

      const void get_TEMPLATE () const;
      void set_TEMPLATE (void value);

      const void get_BASEURL () const;
      void set_BASEURL (void value);

      const void get_VERSION () const;
      void set_VERSION (void value);

  private: //## implementation
    // Data Members for Class Attributes
      void NAME;
      void TEMPLATE;
      void BASEURL;
      void VERSION;

    // Data Members for Associations
      SERVICE *the_SERVICE;
      SERVICE the_SERVICE;
      SERVICE the_SERVICE;
      SERVICE the_SERVICE;
      SERVICE the_SERVICE;
      SERVICE the_SERVICE;
};

// Class WIDL

//## Get and Set Operations for Class Attributes (inline)
inline const void WIDL::get_NAME() const
{
  return NAME;
}

inline void WIDL::set_NAME(void value)
{
  NAME = value;
}

inline const void WIDL::get_TEMPLATE() const
{
  return TEMPLATE;
}

inline void WIDL::set_TEMPLATE(void value)
{
  TEMPLATE = value;
}

inline const void WIDL::get_BASEURL() const
{
  return BASEURL;
}

inline void WIDL::set_BASEURL(void value)
{
  BASEURL = value;
}

inline const void WIDL::get_VERSION() const
{
  return VERSION;
}

inline void WIDL::set_VERSION(void value)
{
  VERSION = value;
}

//## Get and Set Operations for Associations(inline)
inline const SERVICE * WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE * value)
{
  the_SERVICE = value;
}

inline const SERVICE WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE value)
{
  the_SERVICE = value;
}

inline const SERVICE WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE value)
{
  the_SERVICE = value;
}

inline const SERVICE WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE value)
{
  the_SERVICE = value;
```

```
}

inline const SERVICE WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE value)
{
  the_SERVICE = value;
 }

inline const SERVICE WIDL::get_the_SERVICE() const
{
  return the_SERVICE;
}

inline void WIDL::set_the_SERVICE(SERVICE value)
{
  the_SERVICE = value;
 }

#endif
```

참 고 문 헌

[1] Natanya Pitts-Moultis, Cheryl Kirk, "XML Black Book," The Coriolis Group Inc., 1999.

[2] W3C, "Resource Description Framework(RDF) Model and Syntax Specification," August 1998, http://www.w3.org/TR/1998/WD-rdf-syntax-19980819/.

[3] W3C, "Synchronized Multimedia Integration Language(SMIL) 1.0 Specification," W3C, June 1998, http://www.w3.org/TR/1998/REC-smail-19980615/.

[4] "Automating the with WIDL," http://xml .webmethods.com.technology/Automating.html.

[5] W3C, "Web Interface Definition Language (WIDL)," September 1997, http://www.w3.org/TR/NOTE-widl-970922.

[6] "WIDL," http://turtle.ee.ncku.edu.tw/~fencer/WIDL/WIDL.HTM.

[7] 하얀, 황용주, 김용성, "SGML DTD로부터 UML 클래스 다이어그램으로의 사상 알고리즘", 정보과학회논문지(B), 제26권 4호, pp. 508-520, 1999. 4.

[8] 채원석, 하 얀, 김용성, "UML 클래스 다이어그램을 이용한 XML 문서 구조 다이어그래밍", 정보처리 논문지, 제6권 10호, pp. 2670-2679, 1999. 10.

[9] 이미경, 하 얀, 김용성, "RDF 스키마에서 UML 클래스 다이어그램으로의 변환", 정보처리학회 논문지, 제7권 1호, pp. 29-40, 2000. 1.

[10] 채원석, 하 얀, 김용성, "UML 사용사례 및 순서 다이어그램을 이용한 SMIL 문서 동기화", 정보과학회 논문지(소프트웨어 및 응용), 제27권 4호, pp. 357-369, 2000. 4.

[11] Bruce Power Douglass, "Real-Time UML Developing Efficient Objects for Embedded Systems," Addison-Wesley Longman Inc., 1998.

[12] James rumbaugh, Ivar Jacobson, Grady Booch, "The unified modeling language reference manual," Addison Wesley Longman Inc., 1999.

[13] Craig Larman, "Applying UML and PATTERNS: An Introduction to Object-Oriented Analysis and Design," Prentice-Hall, 1998.

[14] E. Akpotsui, V. Quint, C. Roisin. "Type Modelling for Document Transformation in Structured Edition Systems," Mathematical and Computer Modelling, Vol. 25, No 4, pp. 1-19, 1997, http://www.oasis-open.org/cover/.

[15] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl, "From Structured Documents to Novel Query Facilities," In Poc. ACM SIGMOD Intl. Conf. Management of Data, pp.313-324, 1994. 5.

김 상 은

1990년 호서대학교 전자계산학과 졸업(공학사). 1992년 건국대학교 대학원 전자계산학과 졸업(공학석사). 1996년 전북대학교 대학원 전산통계학과 박사과정 수료. 1996년 ~ 현재 대경대학 컴퓨터통신계열 조교수. 관심분야는 SGML/XML, 정보검색, 전문가시스템, 지능형 교수시스템

하 얀

1992년 2월 덕성여자대학교 전산학과 졸업(이학사). 1994년 8월 이화여자대학교 교육대학원 전자계산교육전공 졸업(교육학 석사). 2000년 2월 전북대학교 대학원 전산통계학과 졸업(이학박사). 2000년 9월 ~ 현재 중앙대학교 정보통신연구소 연구전담교수. 관심분야는 XML 응용, 객체지향 모델링, 컴포넌트 모델링, 애니메이션 컴포넌트, 전자도서관, 퍼지 정보검색

김 용 성

1978년 고려대학교 수학과 졸업(이학사). 1984년 광운대학교 전산학과 졸업(이학석사). 1992년 광운대학교 전산학과 졸업(이학박사). 1985년 ~ 현재 전북대학교 컴퓨터과학과 교수. 1996년 ~ 1998년 1월 한국학술진흥재단 전문위원. 관심분야는 디지털도서관, 정보검색, 인터넷 기반 정보 검색, 멀티미디어 시스템, 인공지능 등.