# CORBA기반 분산환경에서 XML문서 교환을 위한 구조변환기법

박민기* · 이재완*

*군산대학교

# The structure conversion mechanism for XML document Interchange in CORBA-based Distributed Environment

Min-gi Park* · Jae-wan Lee**

*Kunsan National University

E-mail : sopiru@kunsan.ac.kr

요 약

분산환경에서 네트워크를 기반으로 하여 정보 문서를 공유하거나 분배함으로써 우리는 원하는 정보를 얻을 수 있다. 하지만 분산환경에서는 시스템의 이종성으로 인해 정보문서 공유 및 분배에 어려움이 있다.

본 논문에서는 정보문서를 효율적으로 처리하기 위해 CORBA 기반의 XML 문서변환 구조를 제안하고, XML 문서변환 구조에 변환기를 설계하여 인터페이스 정의 언어인 IDL을 도큐먼트 타입의 정의인 DTD로 변환하는 기법을 제시한다.

ABSTRACT

In the distributed environment based on the network, we can get the necessary information by sharing or exchanging an information document. However, there is the difficulty for sharing and distributing of the information document because of the heterogeneity at the distributed environment .

In this paper, we propose ) t interchange structure based on CORBA for disposing of the information document and provide the mechanism for interchanging IDL to DTD by designing the converter within it.

키워드

CORBA, XML, IDL, DTD

## Ⅰ. 서 론

인터넷에 대한 기술이 빠르게 발전함에 따라 개방형 표준은 어플리케이션의 미래 가치에 대한 최적의 선택이 되고 있다. 하지만 현존하거나 급부상하는 표준은 극히 적어 분산 오브젝트와 컴포넌트 기술을 근간으로 하고 있는 많은 어플리케이션 서버들은 이러한 표준화를 위해 많은 노력을 하고 있다.

가장 광범위하게 사용되고 배포된 오브젝트 기술은 CORBA이다. CORBA의 핵심은 인터페이스 정의언어인 IDL이다[1].

XML은 ISO 8879에 의해 정의된 SGML의 부분집합이다[2]. XML명세는 2가지 문제를 해결해 준다. 도큐먼트 타입의 정의(DTD)와 도큐먼트 개체를 표현해 준다.

XML은 계층적인 태그 데이터를 표현하는데 중요하고 광범위한 표준이 되고 있다. 이와 더불어 CORBA 인터페이스를 통해 XML 문서들을 처리시키기 위한 노력이 요구되고 있다[3]. 현재 CORBA와 XML의 상호연동을 위해서 DOM API를 사용하여 컴포넌트 기반 기술과 웹과의 호환성을 해결하였다[4]. 하지만 XML이 갖고 있는 수많은 장점에도 불구하고

대부분의 정보문서가 HTTP를 기반으로 하는 문서 형태를 가지므로 이를 기존에 구현된 CORBA 플랫폼에 적용시켜 처리하기에는 어려움이 존재한다.[5]

따라서 본 연구에서는 분산환경에서 CORBA를 기반으로 한 XML 문서 변환 구조를 제안한다. 이 구조의 각 단계별 처리방법은 클라이언트 어플리케이션과 HTTP 웹서버 사이에 HTTP처리방법을 사용하고 XML서비스 구조와 CORBA객체 사이에 IIOP 처리 방법을 사용한다. IDL을 DTD로 변환시키기 위해 XML 문서 변환 구조에는 변환기가 있다. 변환기의 세부 구조는 IDL 트리 빌더와 매핑 프로세스 그리고 DTD 프로세스로 구성된다. IDL 트리 빌더에서는 IDL 파일의 키워드를 IDL 인덱스 테이블을 참조하여 탐색한 후 엘리먼트 트리로 만들고 매핑 프로세스는 엘리먼트 트리에 DTD 요소와 속성에 해당하는 부분을 매핑시킨다. DTD 프로세스는 완성된 DTD를 웹서버에 저장하고 XML 생성자와 XSL 빌더에게 전송하도록 한다.

## II. 관련 연구

### 2.1 인터페이스 정의어(IDL)

OMG IDL(Interface Definition Language)은 구현 객체가 제공하는 인터페이스를 기술하기 위한 언어이다. 이 인터페이스는 명명된 연산들의 집합과 연산에 대한 파라미터들로 구성되어 있다. 또한 IDL에서 만들어진 모든 선언들은 인터페이스 저장소(IR)를 통해서 재 사용할 수 있다. IDL은 객체 지향 개념을 기초로 하고 있으며, 다중상속 및 동적 호출 메커니즘을 지원한다. 이 인터페이스를 이용하여 CORBA 클라이언트는 구현 객체의 서비스를 호출한다. 따라서 OMG는 CORBA 객체의 인터페이스를 정의할 때 반드시 CORBA IDL 언어를 사용하도록 요구하고 있으며, CORBA 표준에 IDL로 작성된 인터페이스를 C, C++, 자바 같은 구현 언어로 변환하기 위한 규칙을 기술하고 있다[6][7].

### 2.2 문서 형식 정의(DTD)

XML은 그 자체로 모든 것을 해결해 줄 수 있는 것이 아니다. 진정으로 필요한 것은 XML 기반의 어휘들 간 서로 상호 연동할 수 있도록 스키마를 구비하는 것이다. 이러한 스키마들 중에는 DTD가 있다. DTD (Document Type Definition)는 문서 생성을 위한 규칙들의 집합으로서, 모든 SGML 문서 작성과 모든 유효한 XML 문서 작성에서 문서 구조를 정의하는데 사용된다[8].

### 2.3 구조 변환 기법

XML 문서들로부터 XML DTD들에 기초한 IDL 밸루타입 계층구조들로 매핑 시키는 설계명세서는 OMG에 의해 발표되었으며 Elenko와 Reinetsen에 의해 XML 과 CORBA 간의 포괄적인 통신 모델이 제안되었고 XML 데이터를 위한 IIOP에 적합한 XIOP를 제안하였다. 하지만 IDL을 XML 스키마로 매핑하는 기법은 아직 발표되지 않은 상태이다[9][10][11].

## III. CORBA기반의 XML 문서변환 시스템

### 3.1 시스템 구조

본 시스템은 분산환경에서 클라이언트와 COR BA 객체간에 표준 문서 처리를 가능하게 하는 어플리케이션 시스템이다. CORBA 플랫폼을 기반으로 한 XML 문서변환 시스템의 구성은 요청을 발생시키는 클라이언트와 도착한 요청을 받아들이고 다시 클라이언트에게 전달하는 HTTP 웹 서버와 웹 서버로부터 넘겨받은 요청에 대해 상호 작용하는 XML 서비스 구조 그리고 이 요청에 대한 응답 메시지를 제공하는CORBA 객체로 구성된다.

각 단계별 처리방법은 클라이언트 어플리케이션과 HTTP 웹서버 사이에서는 HTTP 처리방법을 사용하고 XML 서비스 구조와 CORBA 객체 사이에서는 IIOP 처리 방법을 사용한다.

그림 1은 CORBA 플랫폼을 기반으로 한 XML 문서변환 시스템 구조를 나타낸 것이다.

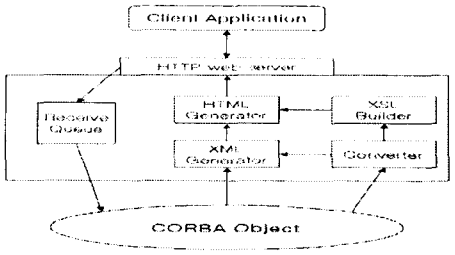


그림 2. CORBA기반의 XML 문서변환 시스템

### 3.2 XML 문서변환 시스템의 구성 요소

XML 문서변환 시스템은 수신큐(Receive Que), 변환기(Converter), XML 생성자(XML Generator), XSL 빌더(XSL Builder), HTML 생성자(HTML Generator)로 구성된다.

클라이언트가 웹서버에게 요청을하면 XML 서비스 구조의 수신큐가 요청을 받는다. 수신큐는 이 요청을 수신된 순서와 우선 처리 요청을 분류하여 순서화 시킨 후 IIOP요청을 한다. CORBA 객체는 IIOP요청을

처리한 다음 IIOP응답을 XML 생성자에게 보내고 변환기에서는 정의된 IDL을 DTD로 변환한다. XML 생성자에서는 CORBA 객체에게서 받은 IIOP응답을 변환기에서 넘겨 받은 DTD를 가지고 DOM API를 통해서 DOM Tree형태로 만든 후 XML 문서를 만든다. 그리고 XSL 빌더에서는 변환기에서 만들어진 DTD에 의해 정의된 추상적 신텍스 트리를 인자화된 출력 규칙을 정의하는 매핑과정을 거쳐 XSL을 만든 후 HTML 생성자에게 보내며 HTML 생성자는 매핑된 XSL과 생성된 XML문서의 개체를 HTML로 표현한 후 최종적으로 클라이언트에게 HTTP 응답을 HTTP 프로토콜을 이용해서 클라이언트에게 보낸다.

### 3.3 변환기 세부 구조

변환기의 세부구조는 IDL 트리 빌더, 매핑 프로세스, DTD 프로세스로 구성되며 IDL 트리 빌더는 코바 객체로부터 전송받은 IDL 파일의 구성요소들을 변환기내에 미리 작성된 IDL 인덱스 테이블을 참조하여 트리 구조로 만든다. 매핑 프로세스는 트리 구조에 DTD 요소와 속성에 각각 해당하는 부분을 매핑 시킨다. DTD 프로세스는 완성된 DTD를 검증하고 저장한 후 XML 생성자와 XSL 빌더에게 전송한다. 그림 2는 변환기의 세부 구조를 나타낸 것이다.

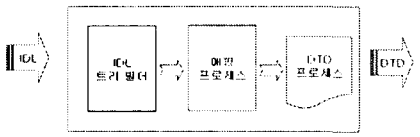


그림 3. 변환기 세부 구조

## IV. 변환기를 통한 XML 문서구조 변환

### 4.1 IDL 트리 빌더에서 IDL의 트리 구조화

IDL 파일은 식별자(module, interface, data type 등의 이름)로 이루어진다. 표1은 IDL 식별자를 나타낸 것이다. interface 키워드는 객체를 정의할 때 사용되며 인터페이스는 attribute와 operation을 갖는다. attribute 키워드는 생략되어서는 안되며 readonly 키워드와 같이 사용된다. interface에 operation을 추가할 때는 method 이름, method 리턴형, parameter 데이터형과 주고받는 방향(in, out, inout), 에러가 발생했을 때에 반환되는 예외 등을 지정한다. module은 관련된 IDL interface를 모으는데 사용된다. 즉, 인터페이스나 데이터 범위를 계층화 하여 분류할 수 있고 IDL 내에 여러 개의 module이 존재할 수 있고 module 내에서 중복적으로 모듈이 들어올 수 있다.

표 1. IDL 식별자

| | |
|---|---|
| module<br>interface<br>attribute<br>mothod<br>parameters<br>param | returns<br>raises<br>exception<br>component<br>..... |

IDL을 DTD의 요소와 속성에 매핑시키기 위한 준비단계로 IDL 트리 빌더에서 IDL엘리먼트 트리 구조로 만든다. 변환기에 미리 작성되어 있는 표 2의 IDL 인덱스 테이블을 참조하여 엘리먼트들의 트리 구조를 작성할 수 있다.

CORBA 객체로부터 IDL을 전달받으면 이진 탐색 기법으로 IDL 내에 있는 키워드를 찾아 나열하고 키워드가 있으면 IDL 참조 테이블에서 인덱스 번호를 가져와 탐색된 키워드에 인덱스 번호를 부여한 후 엘리먼트 트리를 구성한다.

표 2. IDL 인덱스 테이블

| | | | | | |
|---|---|---|---|---|---|
| Module/interface elements | module | 0 | Type definition elements | typedef | 9 |
| | interface | 1 | | constant | 10 |
| Method/attribute elements | method | 2 | | exception | 11 |
| | attribute | 3 | | reference | 12 |
| Struct elements | struct | 4 | | group | 13 |
| | enum | 5 | Descriptive elements | Paragraph | 14 |
| | sequence | 6 | | Note | 15 |
| | union | 7 | .. | .. | |
| | typename | 8 | .. | .. | |

엘리먼트들은 각각의 레코드이며 엘리먼트들과 함께 구성되고 IDL 키워드들을 정의하는 속성, 엔티티등은 필드이며 IDL 키워드는 각각의 레코드를 구분 짓는 키가 된다. 그림 3은 엘리먼트 트리 구성 알고리즘이다.

```
Algorithm for treebuilder
  On receive(IDL<att, method>) from CORBA object
        find <IDL, key> with BINSRCH:
        if (key < > null) then
           get index num from ref_table;
           grant index num to <IDL, key>;
           treebuilder(elm);
        end if
```

그림 3. 엘리먼트 트리 구성 알고리즘

### 4.2 매핑 프로세스에서 IDL트리를 DTD와 매핑

IDL 트리 빌더에서 만든 엘리먼트 트리들은 DTD의 요소와 속성으로 매핑 될 수 있다. 그림 4는 IDL 엘리먼트 트리들을 DTD로 매핑시키기 위한 알고리즘이다.

엘리먼트 트리에서 DTD의 요소(elm)들이 결정되므

로 매핑 프로세스에서는 이 요소들을 전달받으며 DTD 요소와 속성 형태를 선언하여 DTD 요소 형태에 따라 문서의 요소를 작성하고 DTD 속성 형태에 따라 문서 속성을 작성한다. 그리고 하위 요소(sub_elm)를 탐색하여 하위 요소가 요소 안에 존재하면 하위요소를 요소로 변환하여 다음 DTD 요소와 속성을 작성한다. 그림 5는 IDL 엘리먼트들의 트리 구조에 DTD 속성을 매핑시킨 것을 나타내고 있다.

```
Algorithm for mapping <DTD, elm, attr>
 On receive <elm> from elm_tree
  S1: while exist elm do
   DTD elm_format ← <!ELEMENT elm (sub_elm with + or *)
     write (elm) followed by DTD elm_format;
   DTD attr_format ←
     <!ATTLIST elm_name attr_name val_type attr_type "default">
     write (attr) followed by DTD attr_format;
     search(sub_elm);
   if (exist sub_elm within elm) then
      elm←elm[sub_elm];
    go to S1;
   end if
   end
```

그림 4. IDL 트리를 DTD와 매핑시키는 알고리즘

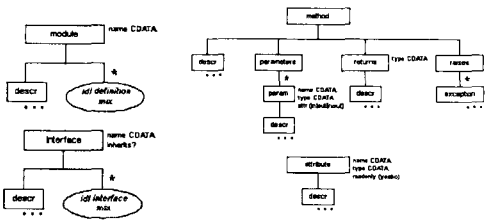


그림 5. IDL 엘리먼트들의 트리 구조와 DTD의 속성 매핑

### 4.3 DTD 프로세스의 DTD 문서구조

매핑 프로세스에서 생성된 DTD를 검증하고 웹서버에 저장한 후 XML 생성자와 XSL 빌더에게 DTD를 전송한다.

그림 6은 IDL내용이 DTD의 요소와 속성으로 정의된 예이다. DTD에서 정의된 요소들은 마크업 언어를 나타낼 수 있는 컨텐트 유형을 결정한다. 요소는 XML DTD나 문서의 중요한 기초 부분이며 속성, 컨텐트 모델, 엔티티, 링크와 같은 나머지 기초 부분은 요소와 함께 구성된다.

```
<!--요소 선언 시작 -->
<!ELEMENT module (interface+)>
<!ELEMENT interface (attribute*, method*)>
<!ELEMENT attribute (readonly*)>
<!ELEMENT readonly (yes|no)*>
.....
<!--요소 선언 끝 -->
```

```
<!--속성 선언 시작 -->
<!ATTLIST module name CDATA #REQUIRED>
<!ATTLIST interface name CDATA #REQUIRED>
.....
<!--속성 선언 끝 -->
```

그림 6. DTD로 정의된 요소와 속성들의 예

## V. 결론 및 향후 연구방향

CORBA는 분산 객체간의 전달 및 요구 바인딩을 위한 프로그램밍 언어에 독립적인 인터페이스 정의 언어(IDL)를 사용한다. 그리고 애플리케이션의 각 서비스들은 미리 정해진 구조를 따르는 적격 XML 문서를 정보 교환용 포맷으로 사용하며 XML 문서의 구조를 결정하는 것은 DTD 이다.

본 논문에서는 CORBA 기반의 XML 문서변환 시스템 구조를 설계하고 그 중에서도 인터페이스 정의 언어인 IDL을 도큐먼트 타입의 정의인 DTD로 변환하는 기법을 보였다. 이러한 개념은 분산환경에서 정보문서를 DTD에 의해 표준화되고 구조화된 전송 포맷으로 규정할 수 있다. 표준화된 정보문서는 신속하고 효율적으로 처리될 수 있으며 구조화된 정보문서는 응용 프로그램이나 공급업체에 구에 받지 않으므로 비즈니스와 전자 상거래를 위한 웹 응용 프로그램 개발에 사용될 수 있다.

그러나 XML을 기술하기 위한 DTD가 XML과 다른 문법을 사용한다는 것은 제대로 된 DTD를 만들기 어렵고 DTD간에 상속 기능이 없는 단점이 있다. 이런 문제를 해결하기 위해 XML 자체의 구문 구조에 기반해서 스키마를 기술 할 수 있는 기법에 대한 연구가 필요하고 추가적으로 본 논문에서 제안한 XML 문서 변환 구조의 성능 분석과 평가가 필요하다.

## 참고문헌

[1] CORBA Specification, Level 1, W3C "http://www.w3c.org"

[2] Frank Boumphrey저, 류광 편역, Professional XML Applications, 정보문화사, pp 30-31, 1999

[3] Document Object Model (DOM) Level 2 Core Specification "http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/"

[4] 이호섭, 홍충선, "분산환경에서의 CORBA와 XML

의 연동을 위한 응용 웹 서버 구조성능 분석", 한국정보과학회 가을 학술발표논문집 Vol.28 No.2, pp 577-579, 2001

[5] Rogue Wave XORBA Link Technical White Paper, 12, 1999 "http://www.roguewave.com"

[6] 구태완, 정연진, 엄상용, 이광모, "RMI-IIOP를 이용한 CORBA 환경에서의 XML 객체 모델링", 한국정보과학회 가을 학술발표논문집 Vol.28. No.2, pp529-531, 2001

[7] George Coulouris, Jean Dollimore, Tim Kindberg, Distributed System: Concepts and Design, 3rd edition, pp 167-169, 2001

[8] W3C XML Specification DTD("XML spec"), "http://www.w3c.org/"

[9] Mark Elenko and Mike Reinertsen, XML & CORBA, Application development tren ds, 9, 1999.

[10] Homepage of XIOP, "http://xiop.sourceforge.net"

[11] XML DOM: DOM / Value Mapping :: ftp://ftp.omg.org/pub/docs/ptc/01-04-04 pdf OMG IDL: Details