스키마 구조 데이타 매핑을 이용한 XML 구조변환 시스템

(An XML Structure Translation System using Schema Structure Data Mapping)

송 종 철 † 김 창 수 †† 정 회 경 †††
(Jong-Cheol Song) (Chang-su Kim) (Hoe-kyung Jung)

요 약 최근의 데이타 처리 환경은 빠르게 변화하고 있다. 특정 단체나, 기업 내에서 상호운용에 대한 고려 없이 서로 다른 목적에 의해서 개별적으로 도입되었던 여러 어플리케이션이나 시스템 등은 유연하고 빠른 처리를 위하여, 프로세스 차원에서 통합되고 연동되어야 할 필요성이 대두되고 있다. 추가 비용을 최소화 하면서 통합에 대한 요구를 충족시킬 수 있는 좋은 방법 중 하나는 모든 플랫폼에서 사용 가능한 장치 비종속적인 데이타 형식이고 W3C(World Wide Web Consortium)의 문서 변환 표준인 XSLT (eXtensible Stylesheet Language Transformation)를 이용하여 필요에 따라 다른 형식의 데이타로 변환이 용이한 XML(eXtensible Markup Language)을 중심으로 통합하는 것이다. 이에 본 논문은 XML 문서의 구조적 정보를 정의하는 XML 스키마(Schema)를 통해 데이타를 제공하는 원본(Source)측과 데이타를 처리하고자 하는 목적(Destination)측의 구조를 나타내고, 이러한 구조정보와 데이타 매핑(Mapping)을 통해 원하는 형태의 구조관계를 정의하며, 정의된 정보를 기반으로 두 구조간의 변환 규칙을 정의한 XSLT 문서를 생성하는 XML 구조 변환 시스템을 설계하고 구현한다. 이렇게 생성된 XSLT 문서를 통해 데이타 처리를 필요로 하는 목적 측의 구조에 맞게 데이타가 재구성 되도록 변환한다. 이렇게, 특정 시스템이나 플랫폼과 관계없이 다양한 구조의 문서를 적용할 수 있게 되고 원하는 형태로의 의미 부여가 가능한 XSLT 문서를 생성하고, 이를 통한 문서간의 변환 처리를 제공하여 데이타의 상호 운용성(Interoperability) 및 확장성을 높이고 XML 문서처리 환경 구축에 기여하는데 목적을 두고 있다.

키워드 : XML, XSLT, Translation, XML Schema

Abstract Last days, various kinds of applications and system were individually introduced into specific groups or enterprises by different objective without considering interoperability among those. However, the environment for data processing is changing rapidly in these days. And now the necessity is growing to integrate and couple applications and system in the process dimension for more flexible and quicker data processing on these application programs and system. When integrating these application programs or system, an integration based on XML is recommended as it is one of good methods which will the additional cost and satisfy the requirements of the integration. This is because the XML is not only device-independent data type which can be used any platform, but also it uses XSLT, the document conversion standard established by W3C, which allows easy data conversion from one to another type on occasion of demands. This paper studies a design and implementation of system to convert XML structure. This system shows the structure of source-side providing data and destination-side processing data with using XML schema that defines structural information of a XML document. And this system defines the structure relationship of desired form as mapping structural information and data. This system creates the XSLT document that defines conversion rule between two structures based information which is defined. The XSLT document which is created as described above will convert data to be appropriate to the structure of the destination-side. By implementing this system, it is able to apply a document into various kinds of structure without considering specific system or platform and it is able to construct XSLT document to which meaning of desired form can

† 비 회 원 : 정보통신연구진흥원 연구원
jcsong@iita.re.kr
†† 비 회 원 : 배재대학교 IT교육센터 강사
ddoja69@mail.pcu.ac.kr
††† 종신회원 : 배재대학교 IT공학부 교수
hkjung@mail.pcu.ac.kr
논문접수 : 2002년 6월 26일
심사완료 : 2004년 6월 15일

be given. This paper aims to offer a process conversion between documents and to improve interoperability and scalability, so that we can contribute to build XML document processing environment

Key words : SAN, Logical Volume Manager, Snapshot, On-line Reorganization

1. 서 론

정보화 사회로 되어감에 따라 컴퓨터의 보급이 늘면서 컴퓨터를 이용한 문서처리의 중요성이 증가해가고 있고, 컴퓨터 관련 분야의 기술진보에 따라 문서의 전자적 처리가 요구되어 이를 위한 워드프로세서, 전자출판 시스템과 같은 전자 문서처리 시스템이 현실화되고 있다. 그러나 이러한 시스템으로 작성된 문서는 각기 독자적인 문서구조와 다양한 내용 정보를 갖고 있으므로 서로 다른 시스템 및 장치를 갖는 문서처리환경에서 이들 문서의 교환 및 공유를 위한 표준 문서구조 모델이 요구되어 왔다. 이에 따라 구조적인 문서의 교환에 관한 필요성이 급증하게 되자 W3C에서는 XML을 새로운 인터넷 표준으로 권고하게 되었다[1,2].

XML의 데이타 교환과 내용 정의라는 장점은 폭넓은 분야에 걸쳐 XML이 활용되고 이의 목적과 특성에 맞는 다양한 XML 응용 표준들이 제정되는 계기를 마련하였다. 이러한 상황에서 XML의 사용범위가 넓어지고, 새로운 시스템으로의 응용도 점차 확대되고 있으며, 또한 새로운 문서로의 변환에 대한 연구도 요구되고 있다. 그러나 이러한 변환의 시도는 각기 다른 구조를 지닌 특정 시스템이나 플랫폼 간의 다양한 표준에 의해 걸림돌이 되고 있다[3].

이에 본 논문은 이러한 이 기종 시스템간의 원활한 데이타 교환 및 확장을 위해, XML 문서의 구조적 정보를 정의하는 XML 스키마를 통해 데이타를 제공하는 원본 측과 데이타를 처리하고자 하는 목적 측의 구조를 나타내고, 이를 통해 원하는 형태의 구조관계를 정의하며, 정의된 정보를 기반으로 두 구조간의 변환 규칙을 정의한 XSLT[4,5] 문서를 생성하는 XML 구조 변환 시스템의 설계 및 구현에 관한 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 XML과 XSLT 및 데이타 매핑에 대한 개요, 3장에서는 XML 구조변환 시스템 설계에 대해 기술하고, 4장에서는 구현 및 고찰에 대해 기술한다. 마지막으로 5장에서 결론 및 향후연구방향을 기술하였다.

2. XML 스키마 및 데이타 매핑

2.1 XML 스키마(Schema)

XML 스키마는 DTD와 같은 개념으로 XML 구조를 정의하기 위한 하나의 표준이다. XML 스키마는 XML 엘리먼트 및 속성을 기술하는데 사용되며, 기본적으로 속성 및 엘리먼트 유형 선언으로 구성되어 있는데 이는 인스턴스 문서 내에서 XML 엘리먼트 및 속성에 관한 내용 모델(content model)을 기술하며, XML 스키마는 상당부분 DTD와 유사한 역할을 수행하지만 DTD보다 더 확장된 기능을 지원한다.

본 논문에서는 W3C의 XML-Data 제안 중 일부인 XDR(XML-Data Reduced) 내에 기술된 XML 스키마를 통해 설계 및 구현되었다[6,7].

2.1.1 XML 스키마와 DTD의 차이점

고유의 구문을 가지고 있는 DTD와 달리, XML 스키마는 XML문법으로 작성되었기 때문에 XML스키마를 작성하기 위한 문법 규칙을 제외한 추가적인 구문을 알 필요가 없다. 그렇기 때문에, XML 문서와 스키마의 공통 구문을 이용할 수 있는 도구가 지원되는 것이 가능하고 XML문서와 스키마는 서로를 지원할 수 있게 된다. 뿐만 아니라, XML문서의 경우 XML 스키마 확장이 가능하여 XML 문서와 마찬가지로 XML 스키마에 엘리먼트 및 속성을 추가 할 수 있으며, 엘리먼트 및 속성이 서로 다른 이름 영역에 존재하는 한, 이들은 스키마 내에서 큰 제약 없이 사용이 가능하다[6].

또한, DTD는 단지 문자열(string) 유형만 지원하는 반면, XML스키마는 정수(ints), 실수(floats), 날짜(dates), 논리 연산자(boolean) 이외에도 다른 간단한 데이타 유형을 다수 지원한다. 만약 정수 값이 요구되는 객체를 다루는 응용 프로그램을 작성하려면, 스키마의 경우 그 값을 바로 사용할 수 있는 반면, DTD의 경우에는 이를 정수로 변환하여야만 할 것이다.

2.2 XSLT 및 XPath

2.2.1 XSLT

XSLT는 문서 변환 언어로서 XML 문서를 다른 문서형태로 변환하거나 재구성하는 역할을 담당한다. XSLT가 XML 문서를 필요한 출력 형태로 변환하는 동안 이루어지는 처리과정은 크게 두 가지로 분리되며 다음과 같다[4,5,8,9].

- 구조적 변환 단계 : 입력된 XML 문서의 구조는 요구되는 출력 형태를 반영하는 구조로 데이타의 변환이 이루어진다.
- 포맷팅(formatting) : 입력된 XML 문서의 구조는 HTML이나 PDF 등의 형식을 가지는 출력 형태를

갖는다.

XSLT에는 스타일 문서의 포맷을 기술할 수 있을 뿐만 아니라, 해당 패턴간의 매핑 규칙의 정의를 포함하기도 한다. 최근에는 임의의 XML 문서에서 다른 구조를 갖는 임의의 XML 문서를 생성하는데 사용되어 지고 있다. 본 시스템은 이러한 XSLT의 특징을 이용하여 설계되고 구현되었다.

2.2.2 XPath

XPath[10]는 XML 문서의 요소와 대응관계를 표현하기 위한 부분으로 XML 문서 구조에 접근하기 위한 언어이다. XSLT를 개발하는 과정에서 문서의 일부분을 선택하는 XSLT의 표현 문법과 문서간의 연결을 위해 개발된 XPointer 언어가 상당히 중첩돼 있다는 사실이 발견되었다. 각각의 위원회는 이러한 표현 언어의 중첩을 피하기 위해 새로운 단일 언어인 XPath를 정의하게 되었다[10].

XPath는 XSLT의 하위 언어로써 그 역할을 수행한다. XPath의 표현은 수치 계산이나 문자열 조작 또는 논리 연산자를 통한 조건문 판단을 위해 사용되지만, 가장 특징적인 용도는 이름이 시사하는 바와 같이 입력 문서 내에서 처리되어야 할 부분을 구분하는 것이다.

2.3 데이타 매핑

매핑은 두 가지의 다른 구조를 지닌 명세 형식에서, 동일하거나 비슷한 의미를 지닌 데이타 요소들을 연결하고, 필요에 따라 연결정의에 추가적인 의미를 부여하여, 두 명세 형식간의 관계를 정의하는 것이다[11].

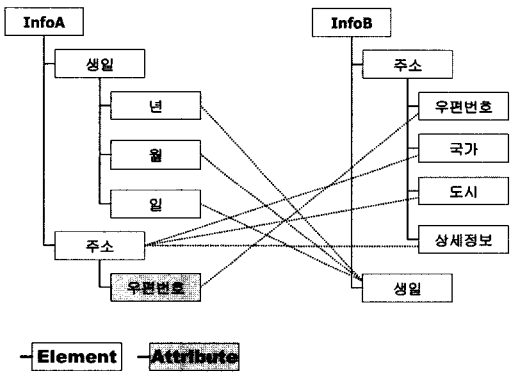

그림 1 두 문서 구조간의 데이타 매핑

그림 1은 두 문서 구조 정보에 대해 데이타 매핑을 통해 정의한 예로써, 보는 바와 같이 두 구조 정보는 같은 내용을 담고 있으면서도 다른 구조를 지니고 있다. 생일이라는 엘리먼트 하위로 년, 월, 일을 구분 짓는 구조를 n:1의 관계로 생일이라는 하나의 엘리먼트에 매핑하고, 주소라는 엘리먼트를 1:n의 관계로 분리하여 매핑하며, 1:1의 관계인 속성으로 표현된 우편번호를 엘리먼트로 표현된 우편번호와 매핑한 모습이다.

본 논문을 통해 구현된 시스템은 XML 스키마로 이루어진 두 문서 구조간의 요소 연결을 통해 상호간의 관계가 정의되고, 이렇게 정의된 관계는 XSLT 문서에 적용되어 각 스키마에 유효한 XML 문서의 변환에 이용된다.

2.4 스키마 데이타 매핑 시 문제점

다양한 형태의 정보자원에 대한 데이타 통합에 대한 요구가 증가하고 있다. 이러한 요구사항에 대해 DTD-TO-Schema 이용한 데이타 통합 및 스키마에 대한 통합에 대한 연구가 이루어지고 있다[12,13]. 하지만 스키마를 통한 정보자원의 통합은 스키마의 이질성에 의한 충돌(conflict)과 데이타간의 이질성에 의해 발생할 수 있는 충돌의 문제가 있다[14,15].

2.4.1 스키마 이질성에 의한 충돌

- 이름충돌 : 스키마에서 사용된 엘리먼트, 속성 등이 같은 이름으로 사용되었으면서도 서로 다른 개념을 표현하거나 서로 다른 개념을 표현하는데 같은 이름으로 사용되는 경우에 발생한다.
- 구조충돌 : 원본 스키마나 목적스키마에서 서로 구조가 다를 때 발생하는 충돌로 엘리먼트로 사용된 구조가 속성이나 다른 구조로 사용될 때 발생한다.
- 타입충돌 : 스키마는 DTD가 문자열만 표현할 수 있는데 반해 다양한 데이타 타입이 표현 할 수 있다. 구조적 의미가 같은 엘리먼트나 속성이 서로 다른 데이타 타입을 가질 때 발생한다.

2.4.2 데이타 이질성에 의한 충돌

같은 의미와 구조로 사용되는 엘리먼트나 속성이 갖는 데이타 값이 서로 다른 의미를 갖거나 다른 제약사항을 갖는 경우에 발생한다.

본 논문에서는 이러한 스키마 충돌에 대해 매핑 정보 관리기를 두어 매핑규칙을 미리 정의하여 매핑규칙을 위배 할 경우 매핑 정의가 이루어지지 않도록 하여 올바른 문법의 XSLT 문서가 생성될 수 있도록 하였다.

3. XML 구조 변환 시스템 설계

XML 구조 변환 시스템은 크게 변환을 위한 두개의 스키마 문서를 입력받아 이를 구조화하기 위한 매핑 요소 입력단계, 두 스키마간의 매핑 정보를 정의하기 위한 매핑 단계 그리고 매핑 정보를 출력하기 위한 매핑 정보 출력단계로 나뉘어 진다. 그림 2는 본 시스템의 전체적인 구성도이다.

본 시스템은 매핑 요소 입력 단계를 통해 변환 주체측인 원본 스키마와 변환 대상측인 목적 스키마 혹은, 이 두 가지 스키마 정보를 모두 포함하는 매핑 정보 문

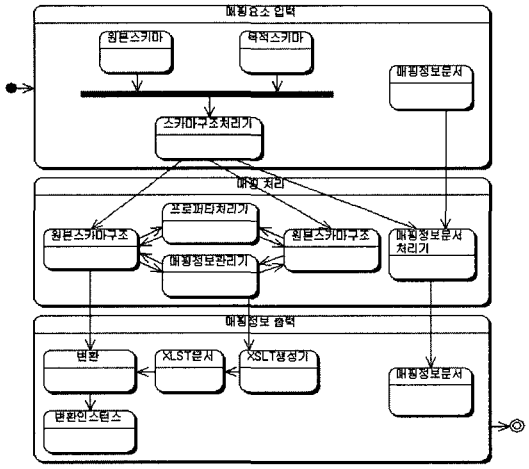

그림 2 시스템 전체 구성도

서의 입력을 통해 이를 논리적 계층 구조인 스키마 구조 트리로 재구성하고 이를 매핑 인터페이스에 적용한다. 매핑 단계에서는 스키마 구조 트리를 통해 매핑하고자 하는 원본과 목적 노드를 연결하고 연결 방법을 설정하여 매핑 정보를 생성하게 된다.

매핑 정보의 출력단계를 통해 매핑 단계에서 생성된 매핑 정보를 토대로 XSLT 문서를 생성하고, 생성된 XSLT 문서를 통한 원본 문서의 변환 테스트 및 매핑 정보가 담긴 매핑 정보 문서의 생성을 처리하게 된다.

3.1 매핑 요소 입력 단계 설계

매핑을 원하는 원본 스키마와 목적 스키마의 입력을 통해 각 스키마 문서의 구조적 정보를 구체화시키고, 매핑의 준비과정을 처리하는 단계이다.

원본 스키마와 목적 스키마를 입력 받아 파서를 통해 이들을 메모리에 로딩하고 기본적인 유효성 검사를 한 뒤 스키마 구조 처리기를 통해 각각의 스키마 구조정보를 트리 형태로 나타내게 된다. 이 단계를 통해 스키마 문서에서 정의해 놓은 각각의 엘리먼트와 속성의 구성과 세부 사항을 쉽게 파악할 수 있음은 물론이고, 매핑 인터페이스를 이루는 기본적인 요소를 제공하게 된다.

3.1.1 스키마 구조 처리기 설계

그림 3은 스키마 구조 처리기의 모습이다. 위의 구조는 스키마 문서의 구조를 DOM(Document Object Model)[16]의 형태로 구성한 것으로써 스키마 문서는 최상위에 스키마 엘리먼트를 두고 있고, 하위로 하나하나의 엘리먼트 및 엘리먼트 하위 정보를 정의하는 여러 개의 ElementType이라는 엘리먼트가 오게 된다[17].

우선 스키마 엘리먼트에 정의된 디폴트(defaut) 이름공간(namespace)을 인지하여, 처리기 내부에서 처리될 모든 요소에 디폴트 이름공간의 이름을 적용한다.

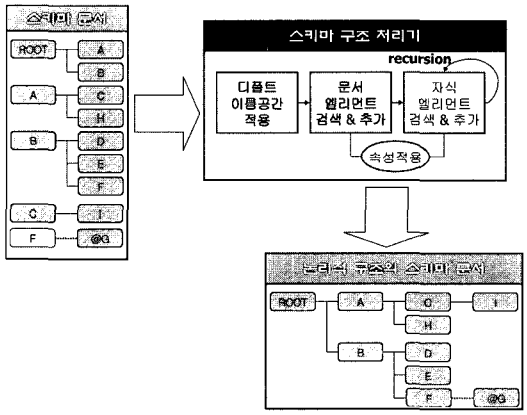

그림 3 스키마 구조 처리기

다음으로 문서 엘리먼트 검색 모듈을 통해 문서 엘리먼트 이름을 인지한 뒤, 여러 엘리먼트 선언을 처리하는 스키마 엘리먼트들 중 *name* 속성으로 해당 값을 갖고 있는 엘리먼트를 검색한다. 검색된 엘리먼트는 최상의 엘리먼트를 정의하는 부분으로써 이 부분을 축으로, 하위에 저장된 엘리먼트와 속성정보는 깊이 우선순위 방식의 탐색 기법에 의해 모두 검색되고 추가되어, 이는 스키마 구조를 나타내는 스키마 구조 트리로 재구성된다.

3.2 매핑 단계 설계

스키마 문서가 로딩 되고 난 후 각각의 스키마 문서 정보를 통해 사용자가 매핑 정보를 정의하는 단계이다. 이 단계에서는 사용자가 쉽게, 원하는 형식의 매핑을 할 수 있도록, 매핑에 필요한 기본적인 정보들과 여러 형태의 매핑을 제공하게 된다.

매핑 단계는 크게 각 노드들의 세부적인 정보를 표현하기 위한 프로퍼티 처리기 모듈과 매핑 인터페이스를 제공하고 매핑 정보를 관리하는 매핑 정보 관리기 모듈로 크게 나뉜다.

3.2.1 프로퍼티 처리기 설계

매핑 시에 기본적으로 참조해야 할 사항은 스키마에 정의된 형식이나 발생 빈도 같은 엘리먼트와 속성의 특성일 것이다. 본 시스템은 이러한 특성을 표현하기 위해 트리 컨트롤 내의 특정 노드 아이템을 선택했을 때에, 하위바의 프로퍼티 탭에 선택한 노드 아이템의 정보가 나타나게 하였다. 이는 프로퍼티 처리기를 통해 처리되며, 그림 4와 같다.

선택한 노드가 최상위 엘리먼트를 나타는지, 그 이외의 다른 엘리먼트를 나타내는지, 속성을 나타내는지를 구분하고, 원본 스키마 구조에서 노드의 위치 정보를 XPath 형태의 구문으로 바꾼 뒤 이를 XPath 구문 검색기를 통해 해당 위치의 노드 객체를 추출한다. 노드의 위치정보는 스키마 구조정보를 트리형태로 나타내어 문

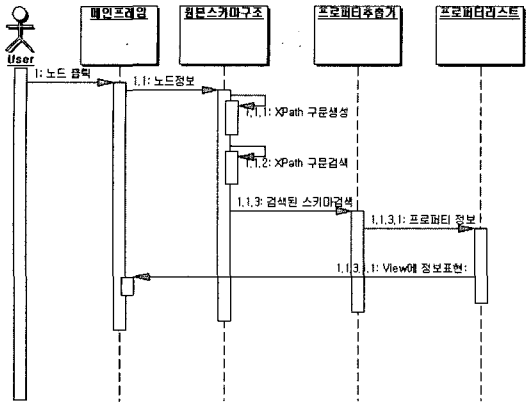

그림 4 프로퍼티 처리기

서객체모델을 통해 최상위 노드를 인식하며, 하위노드는 깊이 우선순위 방식의 탐색기법을 사용하여 반복 처리한다. 추출된 객체는 프로퍼티 추출기를 통해 해당 노드의 상세한 정보를 제공 하게 되며, 이 정보들은 프로퍼티 탭의 리스트 컨트롤에 표시된다. 뿐만 아니라 노드의 형식이나 발생빈도와 같은 세부정보 추출과정의 응용을 통해, 매핑의 중재 및 XSLT 생성기 모듈의 내부 처리에 반영된다.

3.2.2 매핑 정보 관리기 설계

매핑 정보 관리기는 두 스키마 간의 매핑관계 정의를 위한 인터페이스를 제공하고 매핑 중재처리 및 정의된 매핑 정보를 관리하는 역할을 한다.

매핑 정보 관리기에 의해 제공되는 매핑의 형태 및 그에 관련된 설명은 표 1과 같다.

기본적인 매핑의 형태는 위와 같은 경우이며, 다른 기능을 가진 매핑 확장 또한 위와 같은 형태의 범주 내에서 이루어지게 된다.

사용자가 정의한 매핑 정보는 XPath식의 문자열로 매핑 형태에 따라 분류되어 관리되며, 매핑 정보 바를 통해 그 정보들을 확인하고 수정할 수 있다.

그림 5는 사용자가 정의한 매핑 정보의 흐름도이다. 사용자가 매핑을 원하는 각 노드간의 연결을 정의하게 되면, 매핑 중재기를 통해 두 노드의 데이타 타입이나 길이 등과 같은 노드 정보를 비교하고 분석하여, 유효하게 매핑 규칙이 정의 될 수 있는지를 판단한다. 유효하다고 판단되면 이는 매핑 정보 관리기에서 관리되는 매핑 정보 메모리 구조에 추가된다. 이 메모리 구조에 추가되는 정보는 원본측 노드 정보와 목적측 노드 정보, 그리고 매핑 종류를 나타내는 타입 정보와 매핑 종류에 따른 부수적인 정보이며 매핑 정보 객체로서 단일 형태로 관리된다.

매핑 정보는 매핑된 목적 노드 정보에 중심적이기 때문에, 목적 노드 정보들은 동적 배열 형식의 객체에 따로 추출되어 관리된다. 이는 트리 컨트롤 내부에 있는 개개인의 노드 정보를 나타내는 HTREEITEM형으로 기본적으로 정의되기 때문에, 구조적 정보가 중요시 되는 노드 정보의 효율적인 관리가 가능하다. 매핑 정보 객체는 문자열 배열 구조로 관리되며 배열의 인덱스 값에 따라 일반매핑, 대문자매핑, 소문자매핑 으로 정의된다. 올바른 순서가 적용된 XSLT 문서의 생성을 위해 문서 구조 정보 정렬기를 통해 목적 스키마 구조형태에 맞게 정렬되어 다른 배열 객체에 대입되고, 정렬에 의해 바뀐 인덱스에 따라 타입정보 및 원본 스키마 객체 정보 역시 정렬 처리되어 반영된다.

3.3 매핑 정보 출력 단계 설계

매핑 단계에서 생성된 매핑 정보를 토대로 XSLT 문서를 생성하고, 생성된 XSLT 문서를 통한 원본문서의 변환 테스트 및 매핑 정보가 담긴 매핑 정보 문서의 생성을 처리하는 단계이다.

3.3.1 XSLT 생성기 설계

XSLT 문서의 생성은 XSLT 생성기를 통해 이루어지며, 전체적인 처리 과정은 그림 6과 같다. XSLT문서의 골격을 생성해 주는 템플리트 생성기, XSLT 문서의 실질적인 내용인 인스턴스 부분을 생성해 주는 인스턴스 생성기, 함수가 적용된 매핑의 경우에 추가적으로 생성되어야 할 함수 스크립트 부분을 생성해 주는 함수 스크립트 생성기, 이렇게 3가지의 생성기를 통해 전체적인 XSLT 문서 생성이 이루어지게 된다.

이들은 매핑 단계에서 정의된 매핑 정보를 토대로 생성되는데, 원본 매핑 정보에서는 문서의 데이타 정보가

표 1 매핑 형태에 따른 분류

Type Index	매핑 Type	설명	비고
0	일반매핑	일반적인 매핑	1:1
1	대문자 변환 (UpperCase)	원본정보를 대문자화 하여 매핑	1:1
2	문자열 병합 (Concatenate)	여러 개의 원본정보를 결합하여 매핑	n:1 두개 이상의 원본정보 입력
3	부분 문자열 추출 (Substring)	원본정보의 부분 문자열을 매핑	1:n 시작위치 및 문자열길이 값 추가 입력

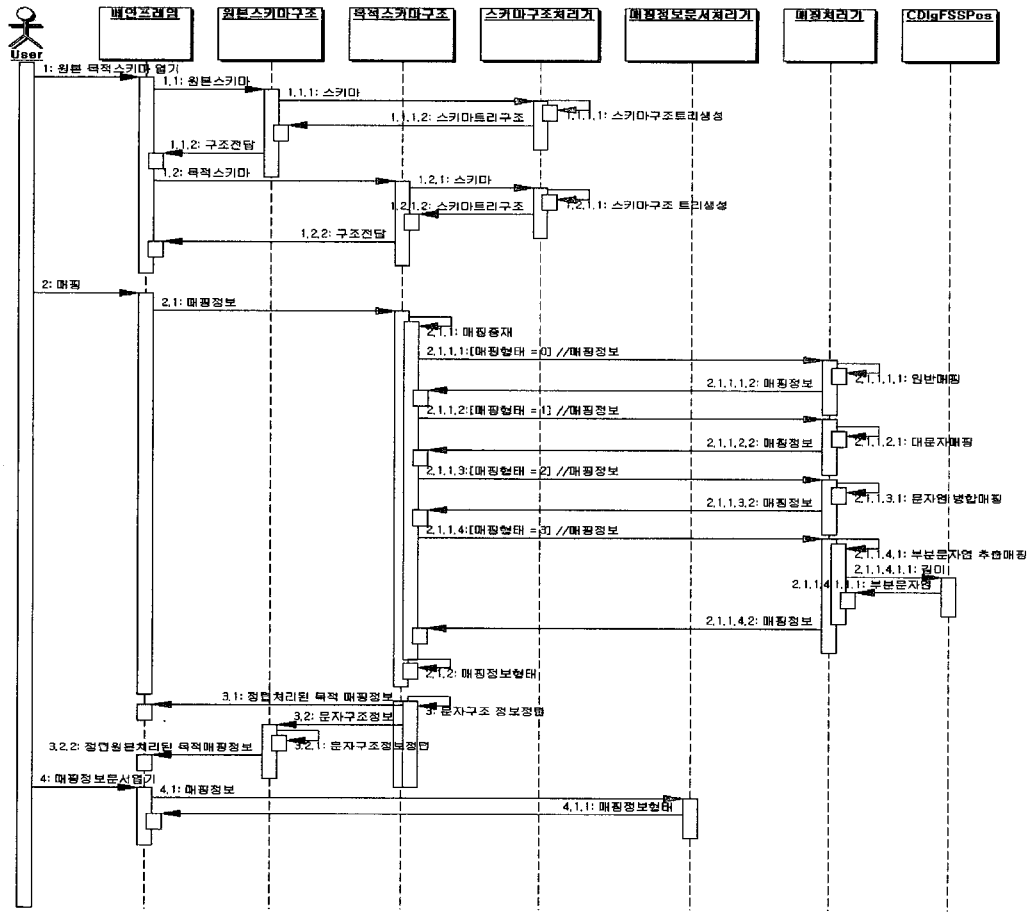

그림 5 매핑 정보의 흐름도

적용되며, 목적 매핑 정보문서는 문서의 구조정보가 적용된다. XSLT 문서의 생성은 매핑과정을 통해 전달된 배열 구조의 매핑형태와 경로정보를 통해 이루어진다. 동일 엘리먼트 단위로 경로정보를 추출하려 동일한 엘리먼트 경로정보를 갖는 집합을 정의하는 배열구조의 동일한 엘리먼트 경로 객체에 전달하고 매핑형태 객체에는 동일한 엘리먼트 경로 객체의 인덱스에 맞추어 해당 타입정보가 대입된다.

템플리트 생성기를 통해 XSLT 문서의 기본 골격을 생성하게 되고, 이러한 기본 골격에 인스턴스 생성기와 함수 스크립트 생성기에서 생성된 인스턴스와 함수 스크립트가 추가된다. 인스턴스 생성기는 매핑 데이타의 반복적 정보와 매핑 형태, 그리고 매핑될 원본 노드가 속성인지 엘리먼트인지의 여부에 따라 각기 다른 인스턴스 구문을 생성하고, 대문자 변환이나 부분 문자열 추출 매핑과 같이 함수 매핑이 정의되어 있는 경우에는 해당 매핑의 형태정보를 함수 스크립트 생성기에게 전달하여 CDATA 형태의 함수 스크립트를 생성한다. 이렇게 생성된 세 개의 요소들이 병합되어 상호간의 변환 규칙이 담겨 있는 XSLT 문서가 생성된다.

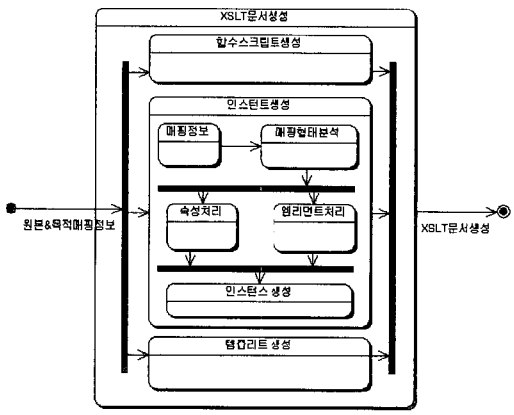

그림 6 XSLT 문서 생성 처리도

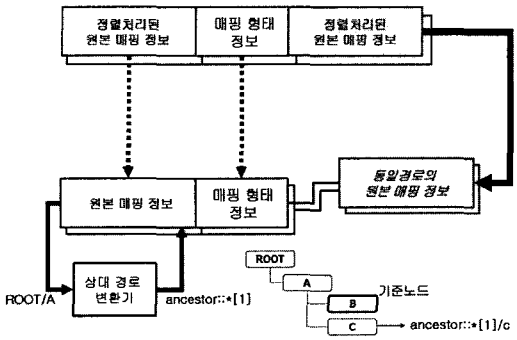

그림 7 인스턴스 생성기

그림 7은 인스턴스 생성기를 나타낸다. 인스턴스 생성기에서는 매핑 정보 관리기를 통해 관리되는 정렬된 형태의 매핑 정보 객체를 가져와서 목적 스키마 노드의 Path 정보를 추출하여 동일한 Path 정보를 갖는 그룹을 정의하기 위한 동적 배열 객체에 이를 할당하고, 해당 목적 스키마 노드에 연결되는 원본 스키마 노드 객체 및 매핑 타입 정보를 함께 할당하게 된다. 이러한 과정에서 하나의 목적 엘리먼트 노드에 연결된 원본 스키마 노드 객체들은 상대 경로 변환기에 의해 원본 노드의 Path 정보는 전체 경로를 표시하는 것에서 특정 노드에 상대적인 경로로 변환되어 관리된다.

그림 7에서 보는 바와 같이 만약 기준 노드의 경로가 'ROOT/A/B'이면서 'ROOT/A' 노드 및 'ROOT/A/C'가 매핑 정보에 포함되어 있는 경우라면 조상 노드를 나타내는 Xpath 구문인 ancestor를 통해 'ROOT/A' 노드의 경우 기준 노드의 1단계위에 있으므로 ancestor::*[1]이라 표현하여 저장하게 되고, 'ROOT/A/C'의 경우에는 ancestor::*[1]/c라 표현하여 저장하게 되는 것이다.

이렇게 생성된 목적 스키마 경로 정보와 이에 연결된 원본 스키마의 상대적인 경로 정보 객체, 그리고 두 객체의 매핑 형식을 정의한 타입 정보를 통해, 엘리먼트 단위별로 인스턴스를 생성하고, 이를 반복 처리하게 된다.

3.3.2 매핑 정보 문서 처리기 설계

매핑 정보 문서는 원본 스키마 정보 및 목적 스키마 정보 그리고 이 두 스키마 간에 정의된 매핑 정보를 모두 담고 있는 XML형식의 문서이다.

매핑 정보 문서의 구조는 그림 8과 같다. 매핑 시에 사용되었던 원본 스키마 및 목적 스키마는 각각 원본 Doc 엘리먼트와 목적 Doc 엘리먼트 하위에 나타나게 되고, 두 문서간의 매핑 정보들은 LinkList 엘리먼트 밑에 나타나게 된다. 이러한 매핑 정보들은 하나의 매핑단위로 link 엘리먼트에 정의된다.

매핑 정보 문서 생성은 매핑 정보문서 처리기를 통해 이루어진다. 이 처리기는 문서 생성뿐만 아니라 생성된

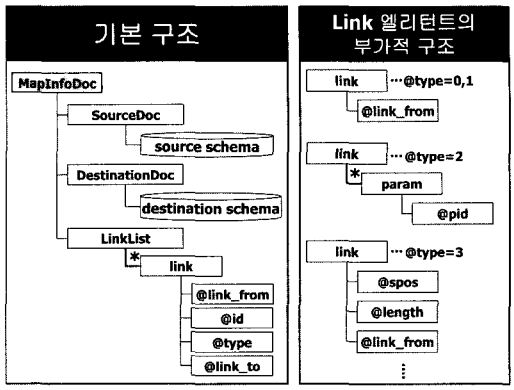

그림 8 매핑 정보 문서 구조

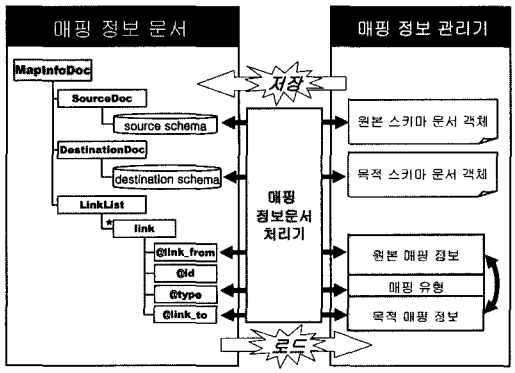

그림 9 매핑 정보 문서 처리기 데이타 흐름도

매핑 정보문서를 통해 매핑 정보의 로딩처리를 담당한다. 이러한 처리는 매핑 정보를 관리하는 여러 객체와 컨트롤들과의 정보 교환을 통해 이루어지는 것으로, 이렇게 생성되고 로딩 되는 매핑 정보문서의 구조와 매핑 정보문서 처리기를 통한 데이타의 흐름은 그림 9와 같다.

3.3.3 문서 변환 처리 과정 설계

원본문서의 변환 처리는 원본 스키마의 구조를 따르는 템플리트 인스턴스를 생성하여 입력하거나 기존의 원본 인스턴스를 입력받아 수행된다. 이러한 실질적인 변환작업은 MSXML 파서 내부의 XSLT 처리기를 통해 처리되며, 변환 문서의 처리 과정은 그림 10과 같다.

원본 템플리트 생성기에 의해 생성되어진 DOM-Document 형의 원본 문서 객체 또는 로딩된 기존 원본 문서 객체와, XSLT 생성기를 통해 생성된 DOM-Document 형의 XSLT 문서 객체를 매개변수로 하는 XSLT 변환 처리 파서 API를 통해 변환처리가 이루어져, 목적 스키마 구조를 갖는 인스턴스로 변환된다. 이러한 처리과정에서는 사용자가 원본측 스키마 구조 정보에 임의의 값을 넣어 원본 템플릿 문서를 생성할 수 있고, XSLT 생성기에서 적용될 XSLT 구문에 임의의

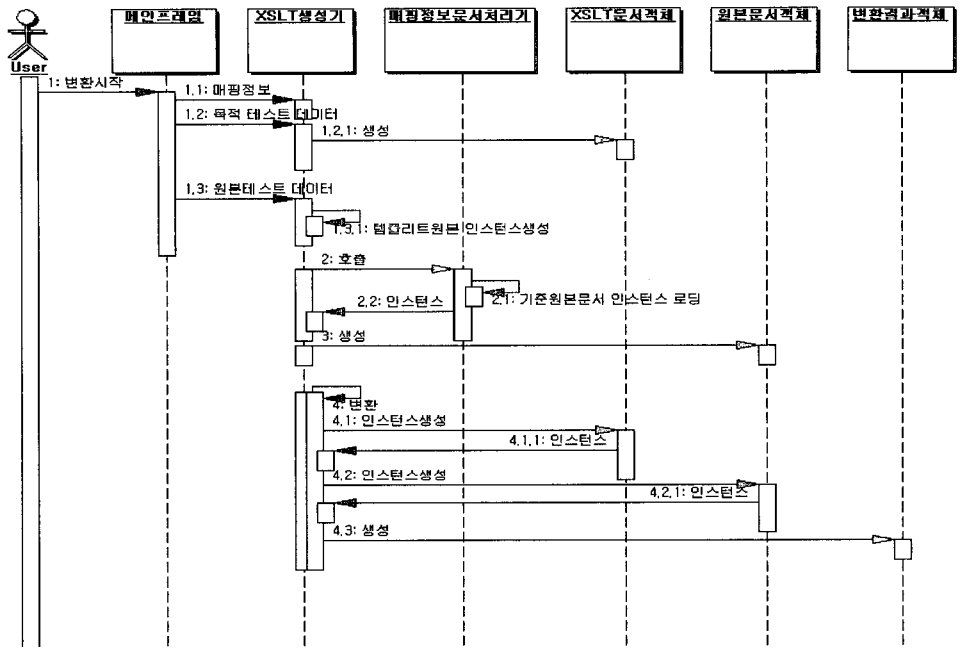

그림 10 변환 문서 처리 과정

목적 스키마 값을 대입할 수 있도록 처리하여, 사용자가 원하는 임의의 결과를 생성 할 수 있는 변환 테스트 인터페이스를 지원하였다.

4. 시스템 구현 및 고찰

4.1 구현

본 시스템은 IBM-PC 호환 컴퓨터(Pentium III-800)에서 개발하였으며, Windows 2000의 운영체제 환경에서 Microsoft visual C++ 6.0 (Developer Studio 98 Enterprise Edition)을 사용하여 구현하였고, 파서 및 XSLT/XPath 처리기는 Microsoft사의 COM(Component Object Model)으로 구성된 MSXML을 사용하였다. 그림 11은 구현된 구조 변환 시스템의 전체 구성도를 보여준다.

그림 11에서 보이는 바와 같이, 본 시스템의 구현은 매핑요소 입력을 처리하고 스키마 구조 처리기를 포함하는 매핑 요소 입력 단계와, 매핑 정보를 생성하기 위한 인터페이스를 제공하고 매핑정보를 관리 하는 매핑 인터페이스 모듈을 포함하는 매핑 단계, 그리고 매핑 정보를 출력하기 위한 XSLT 생성기 및 매핑 정보 문서 처리기를 포함하는 매핑 정보 출력 단계로 나뉜다.

이러한 모듈들은 MFC(Microsoft Foundation Class)를 기본으로 하여, 파서 및 XSLT/XPath 처리기가 포함된 콤 모듈 및 인터페이스 구성을 위한 기타 모듈을 부가적으로 사용한다.

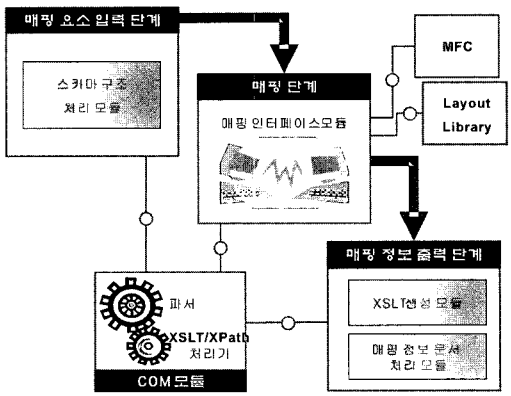

그림 11 시스템 구성도

4.1.1 스키마 구조 처리기 구현

변환 원본과 변환 목적 스키마의 구조화된 객체 트리를 생성하기 위한 과정을 담는 스키마 구조 처리기는 DOM 객체 모듈로써 관리되고 구조적 계층 정보를 표현하기 위해 트리 컨트롤을 통해 표현된다. 이렇게 트리 컨트롤로 표현함으로써 매핑 인터페이스의 기본적인 요소를 제공하고, 사용자가 쉽게 매핑 정보를 정의할 수 있는 환경을 제공한다.

4.1.2 매핑 정보 관리기 모듈 구현

매핑 정보 관리기는 다양한 매핑 인터페이스를 제공하고, 매핑을 위한 중재처리 및 규칙정의를 통해 올바른

매핑이 이루어 질 수 있도록 처리하며 정의된 매핑 정보를 관리하고 편집할 수 있는 환경을 제공한다. 이와 같이 각 노드의 종류에 따른 매핑 가능 여부는 표 2에서 보는 바와 같다.

매핑 하는데 있어 반드시 지키고 기본적으로 알아두어야 할 규칙을 정의하여, 이를 위배 할 경우 매핑 정의가 이루어지지 않도록 하여 올바른 문법의 XSLT 문서가 생성될 수 있도록 하였다. 이를 위해 다음과 같이 기본 매핑 규칙을 정의하였다.

1) 루트 노드에는 매핑이 되지 않는다.
2) 'elementOnly'나 'empty'의 속성을 지닌 element는 'elementOnly'나 'empty' 속성을 지닌 element(또는 'empty'나 'elementOnly'를 지닌 element)와 연결
3) 'textOnly'나 'mixed'의 속성을 지닌 element는 'textOnly'나 'mixed'의 속성을 지닌 element(또는 'mixed'나 'textOnly'의 속성을 지닌 element, attribute에 연결
4) Attribute는 'textOnly'나 'mixed'의 속성을 지닌 element 또는 attribute에 연결
5) 'elementOnly'나 'empty'의 속성을 지닌 element는 N 대 1 매핑이나 1대 N 매 핑을 하지 않는다.

4.1.3 매핑 정보 출력 모듈 구현

정의된 매핑 정보를 저장하고 변환을 위해 XSLT로 출력하기 위한 매핑 정보 출력 모듈은 변환처리에 매핑 정보의 특성을 그대로 반영할 수 있도록 다양한 처리와 구문 생성을 지원한다.

매핑 정보가 적용되는 XSLT를 생성하는 과정에서 가장 중요한 부분인 XSLT 생성기 내부의 인스턴스 생성기는 매핑이 정의된 각 노드 정보의 위치 정보 및 형태 정보를 구분하여 처리하기 때문에, 이 과정에서 매핑의 형태에 따라 인스턴스의 생성도 달라진다.

그림 12에 목적 노드의 형태와 매핑 형태에 따른 XSLT 인스턴스의 생성 구조 및 구조가 적용된 예제를 보인다.

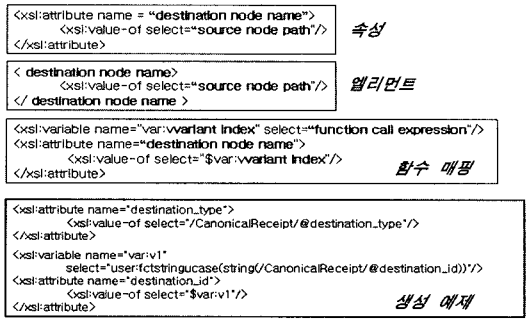

그림 12 목적 노드형태와 매핑형태에 따른 생성구조 및 예제

xsl:value-of 엘리먼트를 통해 매핑된 원본 노드의 값을 가져오고, 목적 노드가 속성이냐, 엘리먼트이냐에 따라 값이 적용되는 방식이 달라지게 되는 것이다. 함수 매핑을 적용하는 경우에는 원본 노드 값 그대로가 아닌 변화된 값을 통해 매핑이 이루어지는 경우이므로, 특수한 목적에 맞는 함수 호출의 표현식이 필요하게 된다. 이를 위해 xsl:variable 엘리먼트를 이용하여 특정함수에 원본 노드 값을 적용하는 함수 표현식을 정의하고 이를 참조하여 XSLT 인스턴스가 생성되도록 한다.

함수 매핑시에 생성되는 함수 스크립트를 생성해 주는 함수 스크립트 생성기의 경우, 대문자 변환과 문자열 병합 매핑을 처리해 주는데 필요한 스크립트를 매핑 정보에 맞추어 생성한다. 그림 13은 함수 스크립트 생성기를 통해 생성된 함수 스크립트와 함수 스크립트를 호출하는 XSLT 문서이다.

문자열 병합 매핑의 경우에는 인자로 받을 수 있는 원본 노드의 수가 정해져 있지 않기 때문에 원본 개수에 맞는 동적인 함수 스크립트의 생성이 필요하므로, 이를 위해 원본의 수를 계산하고 관리하여 함수 스크립트 뿐만 아니라, 함수 스크립트를 호출하는 XSLT 인스턴스 부분까지도 원본의 수에 맞게 동적으로 생성될 수 있도록 하였다.

표 2 각 노드 아이템의 종류에 따른 매핑 가능 여부

원본 노드 형태	가능한 목적 노드 형태
Attribute	Attribute Element (content type : text only) Element (content type : mixed)
Element (content type : element only)	×
Element (content type : text only)	Attribute Element (content type : text only) Element (content type : mixed)
Element (content type : empty)	×
Element (content type : mixed)	Attribute Element (content type : text only) Element (content type : mixed)

```
Function FctStringUCase( p_strA )                    대문자변환
        FctStringUCase = Ucase( p_strA )
End Function

<xsl:variable name="var:v2"
select="user:fctstringucase(string(/CommonInvoice/InvoiceHeader/@Date))"/>

Function FctStringConcat3(p_strParam0,p_strParam1,p_strParam2)
   FctStringConcat3 = p_strParam0 + p_strParam1 + p_strParam2
End Function
                                                     문자열병합
<xsl:variable name="var:v1"
select="user:fctstringconcat3(string(/CommonInvoice/InvoiceHeader/@Type),
string(/CommonInvoice/ReferenceNumber/@Number),string(/CommonInvoice
/ReferenceNumber/@Identifier))"/>
```

그림 13 함수 스크립트 및 해당 함수 스크립트를 호출하는 문서

4.1.4 구조 변환 시스템 구현 결과

그림 14는 시스템의 화면 구성을 보여준다. 화면 양쪽의 원본 스키마 구조뷰와 목적 스키마 구조뷰는 읽어들인 원본 및 목적 스키마 문서를 사용자가 쉽게 인지할 수 있도록 트리형태의 계층적 구조로 보여주며, 트리 아이템 선택 및 드래그 & 드롭을 통한 매핑 인터페이스를 제공한다.

그리고, 각각의 트리 아이템을 선택했을 시, 그에 해당하는 스키마에 정의된 상세한 정보를 원본과 목적 부분으로 나누어진 하위바의 프로퍼티 탭에서 보여주게 된다. 사용자가 매핑 테스트를 위해 정의한 정보는 매핑 테스트 탭에 나타나며, 문서의 처리 결과는 출력 탭에 나타나게 된다.

매핑 정보 바는 사용자가 등록한 모든 매핑 정보를 각각의 매핑 형태에 따라 분류하고, 매핑 정보의 추가 및 삭제 기능을 제공하여 사용자가 매핑 정보를 쉽게 관리할 수 있도록 해준다.

매핑이 가능한 원본 노드 아이템을 드래깅할 시에는 문서모양을 동반한 마우스 커서가 나타나고, 불가능한 원본 노드 아이템을 드래깅할 시에는 매핑 불가능 표시를 나타내는 마우스 커서가 나타나게 된다.

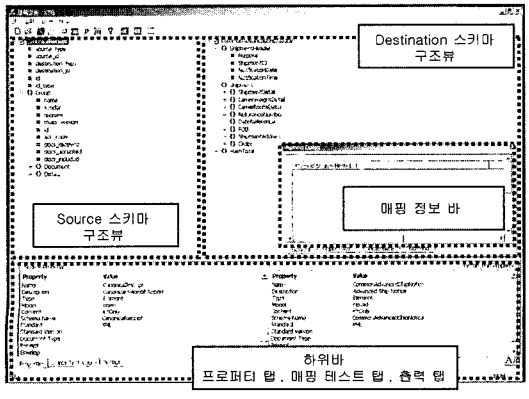

그림 14 시스템 전체 화면 구성

이렇게 드래깅한 마우스 커서를 매핑가능한 목적 노드 아이템에 드롭하게 되면 매핑 팝업 메뉴가 나타나게 되는데, 이를 통해 어떠한 형식의 매핑을 할 것인지 결정하게 된다. 드래그 & 드롭을 통해 구현할 수 있는 매핑은 1:1관계로 이루어지는 일반, 대문자변환, 부분문자열 추출 매핑이며, n:1의 관계로 이루어지는 문자열병합 매핑의 경우에는 다른 방법을 통해 이루어진다. 문자열 병합 매핑은 여러 개의 원본 노드를 선택한 후 매핑을 원하는 노드 아이템 상에서 마우스 오른쪽 버튼을 누르면 나타나는 팝업 메뉴를 통해 매핑 노드 아이템을 선택하게 된다. 이 방식은 문자열 병합 매핑을 포함한 모든 매핑 과정에서 기본적으로 제공된다.

문자열 병합 매핑은 이렇듯 여러 개의 원본 노드 아이템들을 팝업 메뉴를 통해 선택한 뒤 F2를 누르고 팝업 메뉴를 통해 목적 노드 아이템을 선택하여 매핑 정보를 생성한다. 이러한 팝업 메뉴를 통한 모든 매핑과정은 F1키를 누름으로써 취소가 가능하다.

그림 15는 부분문자열 추출 매핑시에 선택된 원본 노드 아이템의 부분 문자열 정보를 설정하는 과정이다. 입력 값은 부분 문자열이 시작될 위치 값과, 부분 문자열의 길이이며, 부분 문자열의 길이를 입력하지 않으면 부분 문자열의 시작될 위치 값부터 선택한 원본 노드 아이템의 문자열 끝부분까지의 부분 문자열을 얻어오는 것으로 인식한다.

그림 16은 매핑 테스트를 실행할 경우에 나타나게 될 특정 노드의 값을 임의적으로 입력하는 모습이다. 값의 입력이 필요한 원본 노드 아이템 상에서 마우스 오른쪽 버튼을 누르면 나타나는 팝업 메뉴의 아이템인 'Input Test Value'를 선택한 뒤, 원하는 값을 입력하면 된다. 이렇게 입력된 값은 하위바의 'Source Test Values' 탭에 등록되며, 등록된 값은 원할 경우 삭제도 가능하다.

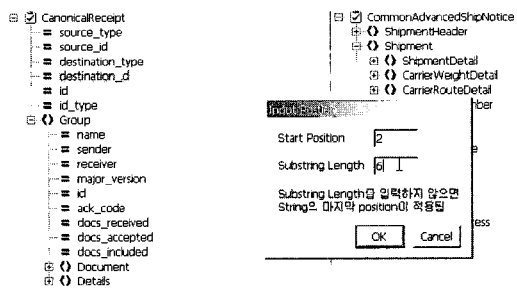

그림 15 부분문자열 추출 매핑에서 시작위치와 길이의 설정 과정

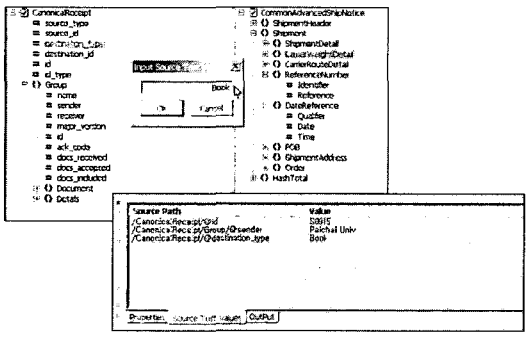

그림 16 원본 테스트 값의 입력과정 및 값이 등록된 모습

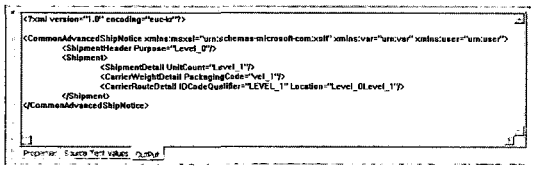

그림 17 테스트 매핑을 실행한 결과

```
<?xml version="1.0" encoding="euc-kr"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:var="urn:var" xmlns:user="urn:user">
  <xsl:output method="xml" indent="yes" omit-xml-declaration="no" encoding="euc-kr"/>
  <xsl:template match="/"> <xsl:apply-templates select="과목별성적"/> </xsl:template>
  <xsl:template match="과목별성적">
    <과목별성적>
      <xsl:for-each select="학생">
        <학생>
          <xsl:attribute name="학생id"> <xsl:value-of select="@학생id"/> </xsl:attribute>
            <xsl:variable name="var:v1" select="user:fctstringconcat2(string(@성),string(@이름))"/>
            <xsl:attribute name="성명"> <xsl:value-of select="$var:v1"/> </xsl:attribute>
            <xsl:for-each select="과목">
              <과목>
                <xsl:attribute name="과목id"> <xsl:value-of select="@과목id"/> </xsl:attribute>
                <xsl:attribute name="과목명"> <xsl:value-of select="@과목명"/> </xsl:attribute>
                <xsl:attribute name="점수"> <xsl:value-of select="@점수"/> </xsl:attribute>
              </과목>
            </xsl:for-each>
          </학생>
      </xsl:for-each>
    </과목별성적>
  </xsl:template>
  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
        Function FctStringConcat2(p_strParam0,p_strParam1)
        FctStringConcat2 = p_strParam0 + p_strParam1
        End Function
    ]]>
  </msxsl:script>
</xsl:stylesheet>
```

그림 18 XSLT문서를 생성한 결과

그림 17은 원본 스키마와 테스트 값을 토대로 하여 생성된 테스트 XML 문서를 사용자가 정의한 매핑 정보를 통해 생성된 XSLT 문서를 통해 변환한 결과이며, 그림 18은 사용자가 정의한 매핑 정보를 통해 XSLT 문서를 생성한 결과이다.

4.2 고찰

본 논문은 변환을 위한 두개의 XML 스키마를 입력받아 매핑 인터페이스를 통해 매핑 정보를 정의하고, 이를 XSLT 문서로 생성하기 위한 XML 구조 변환 시스템의 설계 및 구현에 관한 것이다.

본 논문에서 개발한 구조 변환 시스템은 XML/XSLT 문서의 검증 및 정보 추출을 위해 XSLT/XPath 처리기가 내장된 MSXML 파서를 사용하여 스키마 구조 트리를 생성하였고, 스키마 구조 트리를 통해 매핑 인터페이스를 제공하는 매핑 정보 관리기를 구현하였으며, XSLT형태의 변환 규칙 정보를 제공하는 XSLT 생성기를 구현하였다.

스키마 객체 트리는 표준화된 DOM 인터페이스 사용이 파서에 따라 약간은 상이하므로 이를 위해 C++ 의 이름공간과 컴파일러의 선행처리기를 사용하여 구현하였다. 스키마 객체 트리를 통해 매핑 인터페이스를 제공하는 매핑 정보 관리기는 1:1, n:1, 1:n의 기본적인 매핑 규칙을 프로토타입 형태의 모듈로 구현하였기 때문에 확장이 용이하다. 뿐만 아니라, XSLT 생성 모듈 처리 과정을 최소화하여 빠른 속도로 처리와 생성이 가능하도록 구현하였다.

본 시스템의 장점은 첫째, 매핑 정보를 통해 웹 표준화 기구인 W3C에서 제안하는 XSLT를 생성하고 이를 변환에 응용한다는 점이다. 이로써, 표준화에 입각한 처리 시스템으로써 변화에 능동적으로 대처 가능하고, 데이타의 상호 운용성 및 확장성을 높일 수 있을 것이다. 둘째 매핑 정보를 정의하는 과정과 매핑 정보를 XSLT에 반영하는 과정에서 동작되는 중재역할 처리기를 통해 보다 정확하고 유효한 매핑 정보를 생성할 수 있다는 점이다. 이러한 처리는 사용자에게, 쉽게 확인할 수 있고 쉽게 정정이 가능하도록 하는 인터페이스를 제공하기 때문에 사용자에게 보다 편리하고 정확한 문서의 변환 처리를 제공하게 된다.

단점으로 자식 엘리먼트를 포함하는 엘리먼트와 같이 상위 요소들의 매핑 처리에 대한 설계가 미흡하고, 다양한 역할의 함수가 적용된 매핑지원이 부족한 편이다. 또한, DTD와 같은 다른 문서 구조 정의 표준을 지원하기 위한 추가적인 연구가 필요하다.

4.3 타 구조변환 시스템간 비교

매핑 인터페이스를 기반으로 하여 XSLT를 생성하는 구조변환 시스템은 2000년 초, MicroSoft사의 전자상거래 프레임웍인 Biztalk의 하위 제품군 중 하나로 발표된 Biztalk 매퍼로 부터 시작되었다고 해도 과언이 아니다. 그 이후로 몇몇 응용 제품들이 매퍼라는 부제를 가지고 아직까지도 꾸준히 등장하고 있는데, 이는 데이타 매핑과 XSLT에 기반한 구조변환 시스템이 높은 잠재력과 시장성을 내포하고 있음을 단적으로 말해준다.

본 논문에서 구현된 시스템을 비롯한 모든 구조변환 시스템들은 특정 응용분야에 특성화된 제품이 아닌 이상, 대부분 기본적으로 비슷한 기능과 처리형태를 지닌다. 변환될 XML문서는 사용자가 그 형태를 정할 수 있고 Instance화하여 쉽게 정의할 수 있습니다. 엘리먼트와 속성에 기본값을 지정하는 기능과 함께 여러 번 나오는 엘리먼트를 모두 기술하지 않아도 되는 편리함으로 반복된 작업이 없어 작업효율의 극대화가 가능하며

표 3 타 구조 변환 시스템간 비교

시스템	특징 및 장점	단점
Biztalk Mapper	사용자 중심의 편리한 매핑 인터페이스 제공 E-business에 최적화된 구성요소 제공	한정된 매핑 규칙 정의
XSLT.XML	다양한 형태의 매핑 규칙 정의 기본적으로 가장 충실한 기능제공	불편한 인터페이스
X2X Mapper	DTD입력을 통한 매핑관계 정의 DTD의 특성에 따른 매핑 인터페이스 특성화	복잡한 기능 매핑 규칙 정의와 함수적용 매핑 지원의 부족
본 시스템	확장과 수정에 용이한 프로토타입 매핑 모듈 정의 유효한 매핑 정보 생성을 위한 인터페이스 및 내부 처리 모듈	매핑 규칙 정의와 함수적용 매핑 지원의 부족

InBound와 OutBound형식을 모두 제공하여 단 한번의 클릭으로 두 가지의 Map을 구성할 수 있는 특징을 가진다. 매핑규칙을 분석하여 사용자가 지정한 순서와 횟수에 맞게 데이타를 가져와 XML로 변환 제공하는 과정을 자동으로 수행한다. 본 시스템은 구조변환 시스템으로써 기본적인 역할에 충실할 뿐만 아니라, 프로토타입 매핑 모듈 정의를 통해 XSLT 스펙의 갱신과 기능개선에 따른 확장과 수정이 용이하도록 설계하였다. 그렇기 때문에 WML(Wireless Markup Language) 이나 XHTML(eXtensible Hyper Text Markup Language) 등과 같이 특정화된 문서 포맷을 지원하기 위한 응용프로그램 개발에 쉽게 응용이 가능하다. 표 3에서 대표적인 매핑 인터페이스 기반의 구조변환 시스템간의 특징점 및 단점을 비교한다[18-21].

5. 결 론

컴퓨터의 보급과 더불어 전자문서처리의 요구와 함께 전자문서 교환이 요구됨에 따라 효율적인 전자문서의 관리와 처리를 요구하고 있다. 인터넷의 발전으로 이러한 전자문서의 교환은 급속도로 발전해 왔고 SGML의 장점을 살린 XML은 이러한 문서 교환의 표준으로 자리 잡았다.

그러나 XML을 기반으로 한 다양한 응용프로그램이나, 기업 문서 등과 같이 다른 구조들로 이루어진 여러 표준들 사이에서 문서교환을 위해서는 서로의 구조적 정보에 유효하면서도 한쪽의 구조에 종속되지 않는 독립적인 플랫폼 형태의 처리기가 필요할 것이다.

이를 위해, 본 논문에서는 구조적인 데이타를 표현하는 XML과 이를 정의하는 XML 스키마와 구조적인 데이타 변환 방법을 기술하는 XSLT를 이용하여, 문서 교환 및 정보 공유를 필요로 하는 각기 다른 구조를 지닌 두개의 XML 스키마 문서를 표현하고, 이를 사용자가 원하는 형태로 매핑하여, 이를 토대로 XSLT 문서를 생성하고, 사용자가 정의한 매핑정보를 관리할 수 있도록 하는 XML 매핑 시스템을 설계하고 구현하였다.

본 시스템은 프로토타입 매핑 모듈 정의를 통해 XSLT 스펙의 갱신과 기능개선에 따른 확장과 수정이 용의하도록 설계하였기 때문에, 특정화된 문서 포맷을 지원하기 위한 응용프로그램 개발에 쉽게 활용이 가능하다. 또한 XSLT 생성 모듈 처리과정을 최소화하여 빠른 속도로 처리와 생성이 가능하다는 장점이 있다.

본 연구 결과로 데이타 교환 및 처리에 관련된 다양한 시스템 개발에 많은 영향을 줄 수 있으며, 데이타의 효율적인 통합과 연동을 위한 기반 기술로 유용하게 이용되리라 사료된다.

향후 과제로써, 보다 다양한 매핑 형식의 지원 및 실질적인 응용을 위해 기업이나 특정단체가 지향하는 문서 표준을 XML 스키마로 표현하는 작업, 그리고 매핑 과정의 보다 세부적인 예외 처리 및 확장 처리가 추가되고, 개선되어야 할 것이다.

참 고 문 헌

[1] 정회경 외 2인 공저, "SGML 가이드", 사이버 출판사, 1997.
[2] 정회경, "WWW 문서 작성을 위한 차세대 언어 XML 가이드", 그린, 1998.
[3] W3C, Extensible Markup Language (XML) Version 1.0 (Second Edition), http://www.w3.org/TR/REC-xml, Oct. 6, 2000.
[4] W3C, XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt, Nov. 16, 1999.
[5] Michael Kay, "XSLT Programmer's Reference," WROX, 2000.
[6] W3C, XML Schema Part 0 : Primer http://www.w3.org/TR/xmlschema-0/, May, 2, 2001.
[7] W3C, XML-Data, http://www.w3.org/TR/1998/NOTE-XML-data/, Jan, 5, 1998.
[8] Neil Bradley, "The XSL companion," ADDISON-WESLEY, 2000.
[9] 이형문 배재대학교 석사논문, "XSL-FO를 적용한 XML 문서표현 시스템의 설계 및 구현", 2000.
[10] W3C, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath, Nov. 16, 1999.
[11] Microsoft Biztalk Server, http://msdn.microsoft.

com/biztalk/
[12] C. Baru, A. Gupta, B. Ludascher, R. Marciano, Y. Papakonstantinou, P. Velikhov, V. Chu, "XML-Based Information Mediation with MIX," exhibition program, ACM Conf on SIGMOD'99, 1999.
[13] Won Kim, Jungyun Seo, "Classifying schematic and data Heterogeneity in Multidatabase Systems", Computer, pp.12-18, 1991.
[14] 이강찬, 이경하, 이규철, "XML 기반의 인터넷 정보 자원 통합", 데이타베이스 연구, 한국데이타베이스연구회. 16권 2호, pp.5-21, 2000.12.
[15] 이승원, 권석훈, 김미혜, 이경하, 이규철, "XML Schema를 이용한 스키마 통합시 충돌 문제의 분류", 2001년 한국정보과학회 추계 학술발표논문집(I), pp.31-33, 2001.
[16] W3C, Document Object Model Level 1, http://www.w3c.org/TR/REC-DOM-Level-1.
[17] Charlie Heinemann, "Describe Your Data" http://msdn.microsoft.com/library/en-us/dnexxml/html/xml072099.asp, July. 20, 1999.
[18] mentallink, "http://www.mentallink.com/converter.html"
[19] Tagfree,"http://www.tagfree.com"
[20] Sonic Stylus Studio: Products: Sonic Software, "http://www.sonicsoftware.com/products/additional_software/stylus_studio/index.ssp"
[21] The Castor Project, "http://castor.exolab.org/index.html"

송 종 철

1997년 광운대학교 컴퓨터공학과(공학사). 1999년 광운대학교 컴퓨터공학과(공학석사). 2003년~현재 배재대학교 컴퓨터공학과 박사과정. 1999년~2003 한국전자통신연구원. 2003~현재 정보통신연구진흥원. 관심분야는 Web Service, Semantic Web, 지능형정보검색에이전트

김 창 수

1996년 배재대학교 전자계산학과(학사)
1998년 배재대학교 전자계산학과(석사)
2002년 배재대학교 컴퓨터공학과(박사)
2001년~현재 배재대학교 IT교육센터 책임강사. 관심분야는 멀티미디어 문서정보처리, XML, XML/EDI, XSLT,ebXML, Semantic Web

정 회 경

1985년 광운대학교 컴퓨터공학과 졸업(학사). 1987년 광운대학교 컴퓨터공학과 졸업(석사). 1993년 광운대학교 컴퓨터공학과 졸업(박사). 2001년~2003년 배재대학교 멀티미디어 지원센터장. 1994년~현재 배재대학교 IT공학부 부교수. 관심분야는 멀티미디어 문서정보처리, XML, SVG Web Service,Semantic Web, MPEG-21