웹서비스와 ebXML을 위한 XML 데이터베이스

황태균, 강병영
(동의대)

Ⅰ. 서론

차세대 컴퓨팅 기술의 키워드는 XML 기술을 기반으로 하는 협업(Collaboration)과 통합(Integration)이라 할 수 있고, 향후 eBusiness에 있어서 가장 중요한 개념 중의 하나라고 할 수 있다. 협업과 통합을 위한 다양한 기술들의 표준화가 진행되고 있으며, 이러한 기술들의 핵심 기술은 바로 XML 기반의 웹 서비스를 위한 SOAP, WSDL, UDDI 기술과 eBusiness를 위한 ebXML 기술이다.

실제로, 웹 서비스와 ebXML 기술에서 사용하는 모든 데이터의 표현은 물론 협업과 통합을 위한 모든 기술 역시 XML 기반이다. UDDI 레지스트리(UDDI Registry)와 ebXML 레지스트리/리파지토리 (ebXML Registry/Repository)는 웹 서비스와 ebXML을 위한 주요 기반 기술이라 할 수 있다.

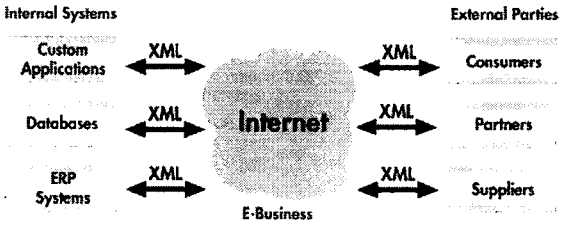
본 논문에서는 차세대 eBusiness의 주요 기반 기술인 XML 데이터베이스의 역할에 대한 연구로 웹 서비스와 ebXML을 위한 XML 데이터베이스의 구성에 대해서 논의하고자 한다.

이 논문을 통해 eBusiness와 XML의 관계는 물론 UDDI 및 ebXML에서의 레지스트리/리파지토리에 대해서도 기술한다.

Ⅱ. eBusiness와 XML의 관계

2.1 eBusiness와 XML

eBusiness는 인터넷과 웹 기술을 사용하여, IT 시스템 내부의 어플리케이션 사이(A2A), IT 시스템과 외부 파트너 또는 공급자의 IT 시스템 사이(B2B), 그리고 IT 시스템과 소비자의 다양한 기기 사이(B2C)의 연결을 구축하는 것을 포함한다. XML은 시스템이 기술자들만이 이해할 수 있는 낮은 차원의 인터페이스뿐만 아니라 현업 종사자들도 이해할 수 있는 높은 수준의 인터페이스도 지원함으로써, 기업 사이에서, 그리고 업무 단위 사이에서 협업이 가능하도록 하기 때문에 오늘날 대부분의 A2A, B2B, B2C 기술의 중심에 있다. 즉, XML이 eBusiness 관계를 구현하는 기반 트랜잭션을 구축하는 동안, 조직은 사람이 이해할 수 있는 친숙한 문서와 동등한 가치를 지닌 전자 정보를 교환할 수 있게 된다.



<그림 1> 시스템과 시장을 통합하는 eBusiness

예를 들면 송장(invoice)과 같은 기업의 비지니스 문서를 디지털 포맷인 XML 포맷으로 정의하고 이러한 XML 포맷으로 기업 사이에서 비지니스 문서를 서로 교환하게 된다. 전통적인 EDI 메지지에 기반한 XML 포맷은 공급망 트랜잭션을 자동화하거나 또는 웹 서비스를 위해 사용된다. 가장 대표적인 메시징 포맷이 바로 SOAP이라 할 수 있다.

XML은 플랫폼과 공급사, 그리고 프로그래밍 언어에 중립적이기 때문에 eBusiness에서 매우 확산되어 있다. XML은 대부분의 기업이 지난 몇 년 동안 지원을 아끼지 않은 서버, 브라우저, 그리고 개발 환경으로 구성되는 기존의 웹 기반 구조를 강화하고 있다. XML의 이러한 다양한 장점들은 기업 문서와 데이터를 교환하는 데에 다시 한 번 네트워크 효과를 유발한다.

2.2 eBusiness의 표준기술 UDDI

UDDI 레지스트리 (UDDI Registry)는 네이티브 XML 데이터베이스(또는 XML 서버)의 특성상 공공(public) 또는 사적(private) UDDI 등기소(registry)로서 서비스하기에 적합하다. XML 서버는 또한 WSDL이나 XML Schemas, XSLT 등과 같은 다른 웹 서비스 메타 데이타의 저장소로

사용할 수 있다. 그리고 ebXML 레지스트리는 스키마 도큐먼트, 비지니스 프로세스 도큐먼트, 거래 당사자들의 협업 프로토콜 프로파일(CPP, Collaboration Protocol Profile), 거래 당사자 간의 협업 프로토콜 약정(CPA, Collaboration Protocol Agreement) 등과 같은 ebXML 거래 당사자들이 거래를 수행하기 위해 필요한 모든 정보를 제공해 주는 곳이다. 따라서 ebXML 레지스트리 서비스는 독립 기관으로 존재하여야 한다.

UDDI 레지스트리와 ebXML 레지스트리에 저장/검색/갱신/삭제되는 모든 데이터는 XML 데이터이다. 따라서, 이러한 UDDI 레지스트리와 ebXML 레지스트리로서 XML 데이터베이스를 쓰고 있다.

- XML이 영속적(persistent)으로 저장될 것을 요구하는 기업 어플리케이션은 일반적으로 적어도 데이터베이스 매니지먼트 시스템(DataBase Management System, 이하 DBMS)이 제공하는 특징들을 필요로 한다. DBMS는 어플리케이션 개발자에게 맡기기에는 너무 중요하고, 너무 어렵고, 너무 값비싼 서비스들을 어플리케이션에 제공하는데, XML 데이터에 대하여도 다른 형태의 데이터와 마찬가지 서비스를 제공하고 있다. 이러한 서비스들은 정교한 트랜잭션 처리, 효율적인 질의, 파일 시스템 수준에서 다루는 것보다 커다란 데이터베이스에 대한 확장성(Salability), 그리고 일상적인 운영을 중단하지 않고도 신뢰성 있는 백업/복구(Backup/Restore)를 제공하는 유틸리티를 포함한다.
- 하나의 웹서비스 트랜잭션은 종종 상이한 기존 기업 시스템에서 여러 개의 동작을 발생시킨다. 예를 들어, 하나의 웹 서비스는 고객관계관리 데이터베이스에 대한 참조, 외부 서비스를 통한 신용도 확인, ERP 시스템을 이용한 주문, 그리고 워크플로우 시스템에의 입력을 동시에 발생시킬 수 있다. 웹 서비스로 전송되는 SOAP 메시지를 native XML 저장소에 기록함으로써 무엇이, 왜 일어났는지를 분명히 기록하고 있는 "감사 기초 자료(audit trail)"를 유지할 수 있을 것이다. 분산된 기간(back-end) 시스템들이 조각조각 나누어 기록하고 있는 트랜잭션에 대한 감사와 분석은 상당히 어렵기 때문이다.
- 기반 시스템의 데이터를 XML로 표현하는 일에는 적지 않은 비용이 들 수도 있지만, 같은 정보가 반복적으로 접근해야 하는 상황에서는 XML 포맷으로 미리 저장해 놓는 것이 효율을 최대화하는 효과적인 방법이 될 수 있다. 이런 상황에서 XML 서버는 훨씬 복잡한 기반 구조에 대하여 분명하고, 훨씬 유용한 뷰(View)를 제공하는 중간 단계(staging) 서버로 기능할 수 있을 것이다.
- 특화된 XML 처리 기능을 위하여 XML 포맷만을 입력 형식으로 받아 들여야 하는 툴이존재한다. 이러한 툴들은 다른 포맷이나 출판 품질 수준의 출력 포맷으로의 변환, XML 마크업 구조, 즉 일반화된 하이퍼링크와 혼합(mixed) 컨텐트 보유 구조의 장점을 살리는 질의 수행, XML 문서의 적절한 구조나 컨텐트를 정의하는 문서의 스키마 또는 DTD에 대한 유효성 검사 등과 같은 기능들을 제공해 주어야 한다.
- XML 기반의 SOAP 프로토콜을 통한 핵심 기업 프로세스에 대한 접근을 제공하는 웹 서비스는 가까운 미래에 대다수 기업의 웹 가능화 전략의 핵심 요소가 될 것이다. 하나의 SOAP 요청은 여러 개의 백-오피스(back-office) 트랜잭션을 일으킬 수 있고, 기존 포맷과 XML을 오가며 변환하는 작업은 매우 지루하고 값비싼 일이 될 것이다. XML 서버에 기존 시스템의 단일한 XML 표현을 유지하는 것은 산재한 기존 시스템의 데이터에 접근하는 것보다 웹 서비스 개발자에게 훨씬 나은 접근성을 제공할 것이다.

2.3 웹 서비스 (Web Sevices)

Graham Glass가 "The Web Services (R)evolution - Applying Web Services to Applications"에서 말하고 있는 웹 서비스에 대한 정의를 살펴보면, 다음과 같다.

"웹 서비스는 다른 프로그램에서 사용할 수 있도록 하기 위해, 하나의 엔티티로 패키지화하여 네트워크 상에 공개된 기능들의 집합이다. 웹 서비스는 오픈 분산 시스템을 생성하기 위한 빌딩 블록이고 회사 또는 개인은 빠르고 값싸게 그들의 디지털 자산을 전세계적으로 사용할 수 있도록 해 준다."

웹 서비스가 오늘날의 기업 업무 형태에 변화를 가지고 올 것이라는 것, 새로운 형태의 어플리케이션이 필요하다는 것이다. 웹 서비스 모델은 수 년 간에 걸쳐 개발된 핵심 경쟁력을 새로운 수

익 창출의 방법으로 사용할 기회를 제공한다. 웹 서비스의 구현과 사용에서 수익성을 담보하는 핵심 요소는 '강력한 통합 기능'과 '공개 표준에 대한 적극적 참여'다.

기술적인 측면에서 보면, 웹 서비스는 표준 인터넷 기술들을 기반으로 하는 동적이고, 분산된 어플리케이션 생성을 위한 상보적 모델이다. 과거의 특정 공급자에 종속적인 기반 구조를 포함하는 종속적 프로토콜에 의지하는 대신, 웹 서비스 사이를, 그리고 사용자와의 접점을 쉽게 연결하는 기반으로 XML과 HTTP를 꼽을 수 있다.

- XML은 이해의 기반이다.
- 웹 서비스를 위한 모든 프로토콜은 이 메타 언어의 문법 기반 위에 있다

웹 서비스는 모든 플랫폼에서 운용 가능하고, 어떠한 프로그래밍 언어로도 구현 가능하다. 이 서비스의 소비자는 서비스가 운용되는 하드웨어나 운영체제에 대해서 아무 것도 알 필요가 없다. 물론 이 서비스를 구현한 객체 모델이나 프로그래밍 언어 역시 알 필요가 없다. 그러나 개별 참여자들의 의사 소통을 위해서 더 많은 표준이 필요하고, 그 예를 SOAP, WSDL, WSFL 등을 들 수 있다. 기반 구조로 웹을 사용하기 위해서는 따라야 할 것들이 분명히 더 있을 것이다.

- SOAP(Simple Object Access Protocol)은 XML을 원격 메쏘드 호출(invocation)과 호출한 원격 메쏘드를 수신자(recipient)에게 전달하는 데에 사용한다. 수신자는 일반적으로 어플리케이션이다.
- WSDL(Web Services Description Language)는 제공되는 서비스를 기술하는 데에 사용된다. 서비스 제공자와 서비스 요청자는 계약 언어로 WSDL을 사용한다. 이것은 서비스 요청자에게 제공되는 메쏘드에 관한 특정 정보를 주고, 따라서 제공자에 대한 링크(연결)이 가능하게 만든다.
- WSFL(Web Services Flow Language)는 여러 개의 웹 서비스들이 하나의 새로운 웹 서비스로 결합되는 지를 기술하는 XML 언어이다. 비지니스 프로세스는 다음 행위를 제공하는 웹 서비스 두 개의 기존 유형으로 이루어진다.

o 사용 유형 : 특정 비지니스 목적을 성취하기 위해서 비지니스 프로세스의 로직을 명시한다. 결과물은 비지니스 프로세스 기술이다.
o 상호작용 유형 : 비지니스 프로세스에 참여한 사람들이 서로에게 제공하는 웹 서비스를 어떻게 연결할 지를 정의한다.

UDDI는 기존의 웹 서비스에 대한 정보를 출판하고, 발견하고, 통합하는 레지스트리(registry)를 제공함으로써 일반적인 웹 서비스 시나리오를 보다 가치 있게 만든다.

- UDDI(Universal Discovery, Description, and Integration)는 찾아보고자 할 전 세계의 비즈니스들에 대한 접근과 비즈니스들의 제품과 서비스에 관한 기술(정보)을 제공한다.

많은 조직이 UDDI를 내부에 소개하는 방식의 하나로, 또는 웹 서비스의 개념과 기술을, 이에 따르는 비즈니스 기업 어플리케이션과의 통합을 테스트하려는 목적으로 내부 UDDI 저장소를 생성하고 있다.

III. UDDI및 ebXML에서의 레지스트리/리파지토리

3.1 UDDI

UDDI(Universal Discovery, Description, and Integration)는 각 비즈니스의 부가정보 및 그 비즈니스에서 제공하는 서비스(API) 정보들을 웹 기반으로 제공해 주는 저장소이다. 여기서 비즈니스란 기업이나 특정 ISP 등 웹 기반의 서비스를 제공해 주는 주체를 일컫는다. UDDI는 특정 제품이 아니라 여러 기업들이 모여서 만든 스펙으로 현재 UDDI 버전 1.0, 2.0, 3.0까지 나와 있다. UDDI 스펙에 대한 보다 자세한 정보는 UDDI 공식 사이트(www.uddi.org)에서 얻을 수 있다.

3.2 UDDI 데이터

UDDI 데이터는 모두 UDDI 레지스트리에서 관리되는데 크게 다음과 같다.

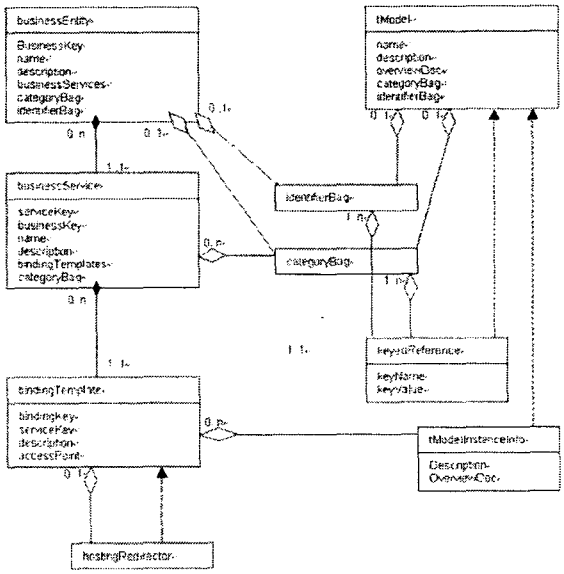
- business Entity : 서비스를 제공하는 비즈니스에 대한 정보. 즉 Service Provider에 대한 정보를 관리한다.

- business Service : 특정 business Entity에서 제공하는 특정 서비스 집합. 여기서는 Service Provider의 세부 서비스(API) 정보를 관리하는 것이 아니라 상위 레벨의 서비스 카테고리 정보를 제공한다.
- binding Template : Service Provider의 실제 Service 정보를 제공. 각 Service에 대해서 설명하는 기술적인 시작점(Technical Entry Point)이다. 여기에는 tModel에 대한 참조 정보가 기술된다
- tModel : 여기서는 Service Provider의 각각의 Service를 기술한다. 정확히 말하면 Service를 기술하는 문서에 대한 참조를 갖고 있다. 예를 들어 WSDL(Web Service Descrition Language) 문서에 대한 HTTP Address 정보가 저장된다.
- publisher Assertion : 비즈니스의 형태가 큰 경우에는 한 기업이 여러 개의 비즈니스(Business Entity)를 가지고 있을 수 있다. 이 경우 여러 개의 business Entity를 묶어서 하나의 publisher Assertion으로 관리한다.

이와 같은 다섯 가지 데이터는 별개로 관리되는 것이 아니라 서로 상관관계를 가진다. 또한 이 데이터들은 XML 형태로 관리되기 때문에 하나의 XML로 표현 가능하다. 예를 들면 다음과 같다.

```
<?xml version="1.0" encoding="utf-8" ?>
<businessEntity>=>business Entity 정보
<discoveryURLs>
<discoveryURL>
http://uddi.microsoft.com/discovery?businessKey=30D0C75C-F423-4AC8-90C5-A00928577B71
</discoveryURL>
</discoveryURLs>
<name>Software AG</name>
<description xml:lang="en">System Software Vendor for Transactional Software, EAI and XML</description>
<contacts>
<contact useType="">
<description xml:lang="en">Director Product Marketing</description>
<personName>Rainer Glaap</personName>
<phone useType="">+49-6151-92-0</phone>
<email useType="">rainer.glaap@softwareag.com</email>
</contact>
</contacts>
<businessServices>=> business Service 정보
<businessService>
<name>Tamino Demo Web Service</name>
<description xml:lang="en">Real Estate Simulation</description>
<bindingTemplates>=> binding Template 정보
<bindingTemplate>
<description xml:lang="en">software ag internal use only at the moment</description>
<accessPoint URLType="http">http://localhost/demoWebSvc/rpcrouter</accessPoint>
<tModelInstanceDetails>
... tModel Info=> tModel 정보
</tModelInstanceDetails>
</bindingTemplate>
</bindingTemplates>
<categoryBag>
<keyedReference/>
</categoryBag>
</businessService>
</businessServices>
<categoryBag>
<keyedReference/>
</categoryBag>
</businessEntity>
```

UDDI 정보 모델(UDDI Information Model)을 구성하고 있는 UDDI 데이터 간의 상관관계는 <그림 2>와 같다.



<그림 2> UDDI 정보 모델의 각 엘리먼트 간의 관계도

3.3 UDDI 프로그래밍 API

UDDI API는 서비스 요청자 또는 서비스 제공자가 UDDI 레지스트리와 연동하는데 사용된다. UDDI API는 크게 공개자 API(publisher API)와 요청 API(inquiry API)로 나뉜다. 공개자 API 는 서비스 제공자가 데이터를 UDDI 레지스트리에 등록하거나 변경하는데 사용되며, add_publisher_Assertions, delete_binding, delete_business 등을 예로 들 수 있다. 요청 API 는 UDDI 레지스트리의 정보를 검색하는데 사용된다. find_binding, get_binding_Detail 등을 예로 들 수 있다. 이러한 UDDI API의 특징은 다음과 같다.

- 모든 API는 SOAP 방식으로 구현된다. 예를 들어 "get_Binding_Detail" 이라는 API는 다음과

같은 SOAP Message로 구현된다.

```
<?xml version="1.0" encoding="UTF-8" ?>
< S O A P - E N V : E n v e l o p e
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelo
pe/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<get_bindingDetail generic="1.0" xmlns="urn:uddi-org:api">
<bindingKey>
B996C07B-A4E2-4DE0-87ED-AF4DD8FE4A32
</bindingKey>
</get_bindingDetail>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

따라서 결과값도 모두 SOAP Message 방식을 따른다.

- SOAP으로 구현되기 때문에 UDDI API는 어떠한 언어로도 구현 가능하다.
- UDDI 레지스트리에서는 등록된 정보에 대해서 등록자 인증(Authorization)을 한다. 이것은 등록한 주체 외의 다른 주체가 등록된 정보를 수정 또는 삭제하는 것을 막기 위해서이다. 따라서 공개자 API의 경우에는 인증 정보가 필요하다.
- UDDI API는 HTTP POST 방식으로 구현된다. 단 공개자 API의 경우에는 HTTPS가 사용된다.
- 모든 UDDI API는 utf-8 방식의 유니코드로 인코딩된다.

3.4 UDDI와 WSDL

UDDI는 원래 웹 서비스에 대한 기술(Description) 정보를 관리하도록 고안되었다. 따라서 WSDL 뿐만 아니라 어떠한 형태의 웹 서비스도 지원한다. 그러나 현실적으로는 주로 WSDL 정보를 관리하는데 사용된다. WSDL 정보는 UDDI의 tModel 에 저장된다. 저장된 예는 다음과 같다.

```
<tModel authorizedName="..." operator="..." tModelKey="...">
<name>StockQuote Service</name>
<description xml:lang="en">
WSDL description of a standard stock quote service
interface
</description>
<overviewDoc>
<description xml:lang="en">
WSDL source document.
</description>
<overviewURL>
http://stockquote-definitions/stq.wsdl
</overviewURL>
</overviewDoc>
<categoryBag>
<keyedReference tModelKey="uuid:C1ACF26D-9672-4404-
9D70-39B756E62AB4"
keyName="uddi-org:types"
keyValue="wsdlSpec"/>
</categoryBag>
</tModel>
...
```

위의 예를 보면 tModel에서는 WSDL 문서의 URL 주소 정보를 갖고 있음을 알 수 있다. 위에서 보여주고 있는 것와 같이 WSDL 문서 자체가 꼭 UDDI 레지스트리에 있을 필요는 없다.

3.5 UDDI의 구현

현재 UDDI 레지스트리를 운영하고 있는 곳은 Microsoft와 IBM이며, Software AG를 비롯한 여러 회사에서 상용 뿐만 아니라 UDDI 테스트를 위한 UDDI 레지스트리를 운영하고 있다. 또한 웹 서비스 툴킷을 제공하는 각 회사에서는 해당 UDDI 레지스트리에 접속하여 웹 서비스를 구현할 수 있는 툴킷을 제공하고 있다. 참고로, Software AG에서 운영하고 있는 Tamino 개발자 커뮤니티(Tamino Developer Community)에서 제공하는 UDDI 데모 레지스트리는 다음과 같다.

3.5.1 UDDI 와 WS-I(Web Service Inspection Language)

UDDI는 모든 웹 서비스 정보를 소수의 UDDI 레지스트리에서 관리한다. 이 경우 레지스트리 입장에서는 관리 문제가 발생하고 사용자 입장에서는 입력 또는 검색하는데 오버헤드가 발생할 수 있다. 이러한 구조적인 단점을 보완하기 위해서 MS와 IBM이 만든 것이 WS-I이다. WS-I의 가장 큰 특징은 웹 서비스에 대한 정보를 별도의 레지스트리에서 관리하는 것이 아니라 각각의 서비스 제공자에서 제공한다는 것이다. 따라서 서비스에 대한 정보를 서비스 제공자가 직접 제공한다. WS-I를 사용했을 경우 장단점은 다음과 같다.

	UDDI	WS-Inspection
장점	- 강력한 데이터 검색 기능 지원 - 여러 서비스 제공자의 정보를 검색 가능	- 시스템 오버헤드가 적음
단점	- 시스템 오버헤드가 많음	- 상대적으로 데이터 검색기능이 약함 - 상대방의 서비스 정보만 검색 가능

MS와 IBM에서는 WS-I가 UDDI의 단점을 보완한 상호보완적 스펙이라고 설명한다. 즉 규모가 큰 웹 서비스의 경우에는 UDDI를 사용하고 규모가 작은 웹 서비스의 경우에는 WS-I을 사용하는 것이 좋다는 것이다.

3.6 ebXML

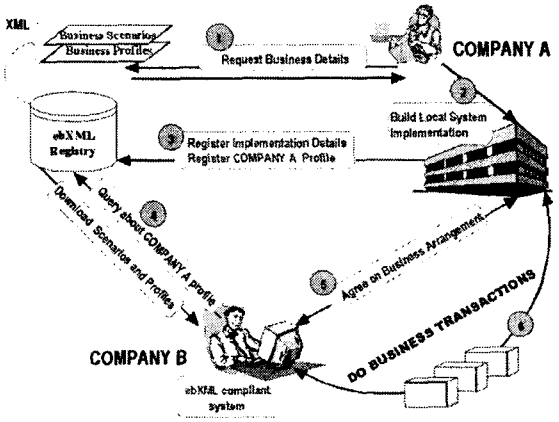
UN/CEFACT와 OASIS는 2001년 5월 14일 Vienna 회의에서 ebXML Phase 1을 발표했다. ebXML 의 목적은 전 세계적인 규모의 상거래를 XML 기반으로 추진하기 위한 기술적인 기반을 만들어 놓는 것이다. 이 작업을 위해서 1999년부터 18개월간의 작업을 거쳐서 ebXML이 만들어지게 되었다. ebXML은 단순히 XML 언어의 스펙만을 규정한 것이 아니라 비지니스 프로세스(Business Process)의 설계에서부터 통신 방식에 이르기까지 다양한 분야를 다루고 있다. ebXML에 대한 관련 정보와 모든 스펙은 ebXML 공식 사이트(http://www.ebxml.org)에서 다운로드 받을 수 있다.

기존에도 범세계적인 무역을 위한 여러 가지 규약이 있었으나 그 규약들은 XML 기반이 아니었고, 모든 업종을 포괄하는 것이 아니었다. 또한 시스템 구축 시 비용이 많이 드는 어려움이 있었다. 따라서 ebXML은 다음을 목표로 만들어졌다.

- W3C XML을 기반으로 구축
- ebXML 을 지원하는 거래 당사자들간의 상호 연동성, 효율성을 극대화
- 기존의 EDI 등의 시스템으로부터 전환이 용이하게 구성
- 국제 기관에서 표준을 제정, 관리하여 특정 제품에 종속적이지 않도록 함
- B2B, B2C 등 기존의 모든 거래 종류 및 큰 기업에서부터 작은 기업까지 모든 기업에서 적용할 수 있는 표준을 제정

시나리오는 테크니컬 아키텍처(Technical Architecture) 에서 설명하는 ebXML의 간단한 시나리오이다.

<그림 4>는 A사와 B사가 ebXML을 수행하는 과정을 그린 그림이다. 여기서 ebXML 레지스트리는 ebXML 당사자들에 대한 정보를 관리하는 XML 기반의 데이터베이스이다. ebXML 레지스트리는 반드시 A사나 B사에 속해 있을 필요가 없으며, 오히려 독립적으로 운영되는 기관으로 보는 것이 더 현실적이라 할 수 있다.



<그림 4> ebXML 시나리오

1) A사의 담당자는 ebXML 레지스트리를 검색하여 자기가 원하는 비즈니스 파트너가 있다는 것을 알게 된다. 이제 A사는 ebXML 을 사용하여 비즈니스를 수행하기로 결정한다.
2) 따라서 ebXML에서 규정하고 있는 여러 가지 기술적 사항들을 구현한 시스템을 구축한다.
3) 시스템을 구축한 후 ebXML Registry에 자신의 CPP를 등록한다. 여기서 협업프로토콜프로파일(CPP, Collaboration Protocol Profile)은 ebXML을 수행하는데 필요한 자사의 정보를 XML 형태로 기술한 문서이다. 자사 정보, 자신이 수행하는 비지니스 프로세스(Business Processes), 비지니스 서비스 인터페이스(Business Service Interface; 사용하는 통신 프로토콜, 보안 기술 등) 등의 내용을 다루고 있다.
4) B사는 ebXML 레지스트리를 검색하여 자신이 원하는 비지니스 파트너로서 A사가 있다는 것을 알게 된다. 여기서 B사는 이미 레지스트리에 자사의 CPP를 등록했다고 가정한다.
5) B사는 A사에 접속하여 거래 성사를 위한 CPP를 서로 교환하게 되고 여기서 협업프로토콜약정(CPA, Collaboration Protocol Agreement)가 만들어진다. CPA란 두 회사가 실제로 거래를 수행하기 위한 규약으로써, 내용 항목은 CPP와 비슷하나 내용은 두 CPP간의 공통 부분을 기반으로 구성되어 있다. 또한 이 단계는 사람의 손을 거쳐서 이루어지는 것이 아니라 서로의 CPP에 기반하여 자동으로 CPA를 구성하는 것을 ebXML이 지향하고 있다.
6) 앞에서 만들어진 CPA를 기반으로 두 회사 간에 거래를 수행한다.

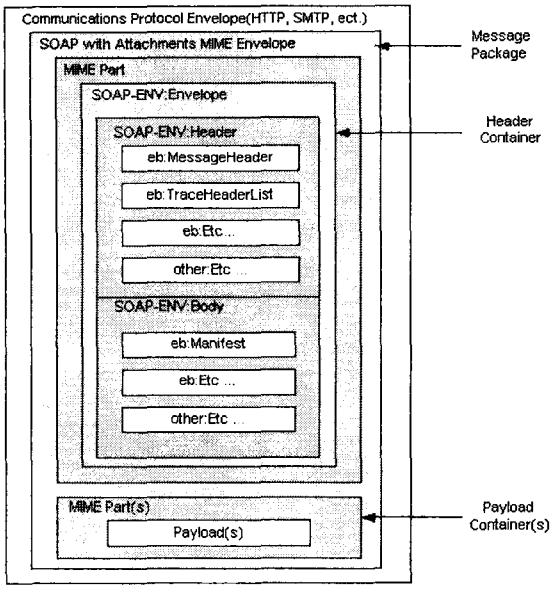
위의 예는 일대일 당사자간의 간단한 거래를

예로 들었으나 ebXML은 일대일뿐만 아니라 다대다 등의 복잡한 거래에서도 적용될 수 있게 만들어져 있다.

3.7 ebXML 메시지 서비스 스펙

ebXML 기반의 모든 데이터 교환은 "ebXML Message Service Specification"에서 규정한 방식으로 이루어진다. 즉 자사 내에서가 아닌 모든 거래 즉, 기업-레지스트리 서비스, 기업-기업 등 모든 온라인 상의 데이터 교환은 이 스펙을 바탕으로 이루어진다.

ebXML 메시지 서비스 스펙은 SOAP(Simple Object Access Protocol) 을 기반으로 구성되어 있으며, 실제로 ebXML에서는 원래의 SOAP 스펙을 확장한 "ebXML SOAP Extensions"을 만들어서 사용하고 있다. 다음은 ebXML의 메시지 구조이다.



<그림 5> ebXML 메시지 스펙

위의 구조에서 SOAP-ENV:Envelope 이하의 부분은 SOAP Extension이고, 별도로 Payload 부분을 지정해서 XML 포맷이 아닌 데이터 (이미지, 텍스트 등) 도 전달할 수 있게끔 하고 있다. Message Service의 특징은 다음과 같다.

- ebXML Message는 통신 프로토콜에 독립적이다. 따라서 HTTP 뿐만 아니라 SMTP 등 텍스트 형식의 데이터(Multipart형식)를 보낼 수 있는 어떠한 프로토콜로도 구현 가능하다.
- ebXML Extension에서는 기존의 SOAP에서 규정한 태그는 그대로 수용하면서, 확장 가능한 부분에 ebXML 태그들을 추가하여 사용하고 있다. 위에서 보듯이 ebXML 확장 태그들은 "eb" 네임스페이스를 사용하고 있다.
- ebXML Message는 ebXML의 모든 Spec을 다 지원한다. 또한 각 Spec에서는 데이터나 메세지를 설명하는 부분에서 Message Service와의 연관관계, 구현방법 등을 기술하고 있다.
- Message Service에서는 단순히 메세지의 구조뿐만 아니라 거래 당사자간의 인터페이스 방식도 규정하고 있다. 예를 들면 상대방의 상태를 검사하는 Status Request, 신뢰성 있는 통신을 위한 Reliable Messaging, Error Reporting, Security 등의 부분도 규정짓고 있다.

3.8 ebXML 레지스트리

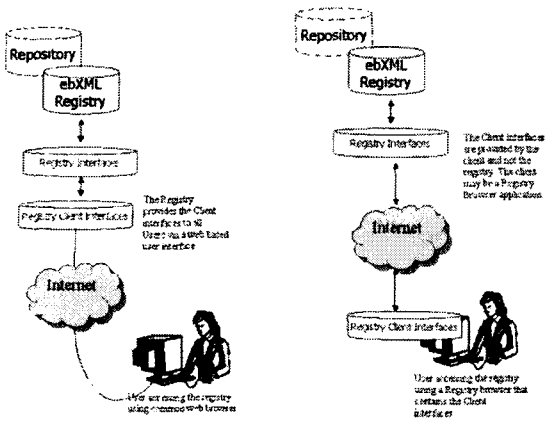
ebXML에서는 ebXML 레지스트리 서비스 (Registry Services) 및 ebXML 레지스트리 정보 모델(ebXML Registry Information Model)을 각각 정의하고 있다. ebXML 레지스트리는 ebXML 거래 당사자들이 거래를 수행하기 위해 필요한 모든 정보를 제공해 주는 곳이다. 따라서 ebXML 레지스트리 서비스는 독립 기관으로 존재할 가능성이 크다. ebXML 레지스트리 서비스에서 제공하는 데이터는 다음과 같다.

- 스키마 도큐먼트(Schema Document) : 업종별 표준 스키마로서, "Technical Architecture Spec"의 "Core Component" 항목에서 설명함.
- 비지니스 프로세스 도큐먼트(Business Process Document) : 비지니스 프로세스 명세 스키마(Business Process Specification Schema)에서 만들어진 문서
- 거래 당사자들의 CPP
- 거래 당사자간의 CPA

위의 데이터 뿐만 아니라 ebXML을 통해 이루어지는 모든 데이터는 ebXML 레지스트리 서비스에서 관리될 수 있다. 레지스트리 서비스의 구조는 다음과 같다.

위에서 보면 ebXML 레지스트리 서비스는 다음과 같이 크게 네 부분으로 이루어짐을 알 수 있다.

- 레지스트리(Registry) : 레지스트리 서비스를 제공하는 핵심 엔진
- 리파지토리(Repository) : 서비스하는 실제 데이터를 저장, 관리하는 엔진

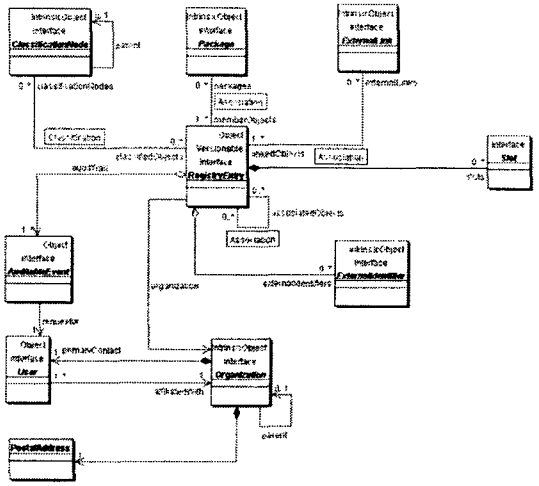


<그림 6> 레지스트리 서비스의 구조

- 레지스트리 인터페이스(Registry Interfaces) : 거래 당사자들이 레지스트리 서비스를 이용할 수 있도록 서버 측면에서 제공하는 인터페이스. 프로그래밍 관점에서 보면 서버 API라고 할 수 있다. 이 인터페이스를 통해서 데이터를 입력, 조회, 관리할 수 있다.
- 레지스트리 인터페이스(Registry Interfaces) : 거래 당사자들이 레지스트리 서비스를 이용할 수 있도록 서버 측면에서 제공하는 인터페이스. 프로그래밍 관점에서 보면 서버 API라고 할 수 있다. 이 인터페이스를 통해서 데이터를 입력, 조회, 관리할 수 있다.

이 네 가지 항목 이외에도 거래 당사자들이 정보를 조회할 수 있도록 SQL 쿼리(SQL Query) 기능도 선택 항목으로 제시하고 있다. ebXML 레지스트리 서비스의 특징은 다음과 같다.

- 레지스트리 서비스는 특정 DBMS에 종속적이지 않다. 즉 레지스트리 인터페이스만 지원한다면 어떤 형태로도 적용 가능하다.
- 레지스트리와 리파지토리는 기능 측면에서 구별된다. 레지스토리는 데이터에 대한 메타데이터를 관리하고 인터페이스를 제공하는 반면 리파지토리는 실제 데이터를 가지고 있다. 구현 측면에서 레지스토리가 반드시 리파지토리를 가지고 있어야 하는 것은 아니다. 단 이 경우에는 다른 곳에 있는 레지스트리를 통해서 데이터를 제공할 수 있어야 한다.
- 레지스트리간의 데이터 연동이 가능하다. 즉 데이터를 여러 곳에 분산해 놓고 서비스를 제공할 수 있다.
- 레지스트리 내의 모든 데이터는 유일한 UUID(Universally Unique Identifier)를 부여받는다. 이로 인해서 모든 데이터를 유일하게 추출할 수 있다.
- 레지스트리 내의 데이터 구조에 대해서는 RIM(Registry Information Model)에서 규정하고 있다. 다음은 RIM의 전체 구조이다.



<그림 7> ebXML 레지스트리 정보 모델

위의 모델은 레지스트리 메타 데이터를 설계한 것이다. 즉 실제 데이터에 대한 설계는 RIM에서 설명하고 있지 않다. 모든 RIM은 UML 표기법으로 표기하였다. 이것은 단순히 스펙를 표현하기 위한 방법이고 ebXML을 구현하는데 꼭 UML을 사용해야 하는 것은 아니다.

- RIM에 대한 접근은 레지스트리 인터페이스를 통해서 이루어진다.

Tamino XML Server와 같은 XML 데이터베이스를 이용하여 레지스트리 및 리파지토리 서비스를 제공하기 위한 DBMS로 활용 가능하다. 특히 모든 데이터가 XML 포맷으로 만들어져 있기 때문에 Tamino XML Server에서 제공해 주는 XML 서버 고유의 기능을 활용하여 보다 쉽고 빠르게 레지스트리 및 리파지토리를 구축할 수 있다. 한 가지 주의할 점은 SQL을 지원하는 부분은 별도의 관련 모듈을 추가해야 서비스가 가능할 것이다.

Ⅳ. 결론

향후 웹 서비스와 같은 표준 기술을 이용하여 모든 엔터프라이즈 내의 어플리케이션이 통합되고, 웹 서비스 및 ebXML 등과 같은 표준 기술을

이용하여 기업 간 비즈니스가 통합되면, 통합 시스템에 대한 유지보수 문제가 발생하게 될 것이다. 또한, 이미 구축되어 운영 중인 시스템을 구성하고 있는 컴포넌트 형태의 소프트웨어 모듈을 교환하거나 업그레이드하기 위해서는 컴포넌트에 대한 배치 문제 역시 심각하게 고려해 주어야 할 것이다.

따라서, 본 연구에서는 차세대 eBusiness의 주요 기술인 웹서비스 기술과 XML 데이터베이스의 역할을 분석하기 위해 웹 서비스와 ebXML을 위한 XML 데이터베이스의 구성에 대해서 연구해 보았다. 본 논지를 통해 eBusiness와 XML의 관계는 물론 UDDI 및 ebXML에서의 레지스트리/리파지토리에 대해서 향후 계속적인 연구가 진행되어야 할 것이다.

참고문헌

[1] 산업자원부, 한국전자거래진흥원, "2003 e-비즈니스 백서", 2003

[2] 산업자원부, 한국전자거래진흥원, "2002 ebXML 백서", 2002

[3] 산업자원부, 한국전자거래진흥원, "2004 e-biz 표준화백서, 2004

[4] XMLOne 개발자 네트워크 http://www.xmlone.co.kr/ttn/jsp/ttnbody.jsp

[5] UDDI 공식 사이트 http://www.uddi.org/

[6] ebXML 공식 사이트 http://www.ebxml.org/

[7] IBM UDDI 홈페이지 http://www-306.ibm.com/software/solutions/webservices/uddi/

[8] IBM Web Service Toolkit http://www.alphaworks.ibm.com/tech/webservicestoolkit

[9] Microsoft UDDI 홈페이지 http://uddi.microsoft.com/

[10] EbXML, JavaTM Technology, and Web Services TS-3380, Java One 2002 http://www-106.ibm.com/developerworks/webservices/library/ws-peer1.html?dwzone=ws