동적 비즈니스(Dynamic Business) 실현 예고

박범대 | 한국전산원 지식정보기술단 전자거래연구부 주임연구원

1990년 대 말 XML(eXtensible Markup Language)의 출현으로 이를 기반으로한 여러 전자거래 프레임워크들이 등장하게 되었다. 대표적인 것들이 eCo 프레임워크나 RosettaNet, 그리고 최근까지 전자거래 분야에서 국제적 관심을 이끌었던 ebXML 등이 있다. ebXML은 1999년 11월부터 현재까지 표준화 작업을 진행하고 있지만 비즈니스 프로세스 및 핵심 컴포넌트 부분의 표준화 작업의 진전이 더딘 관계로 잠시 주춤한 상태이다. 이 틈새로 웹 서비스(Web Services)가 급부상하게 되었고 이를 지원하는 솔루션들이 일부 나오기 시작하면서 그 실현 가능성에 목소리를 더 높이고 있다.

사실 웹 서비스는 태생 자체가 전자거래를 배경으로 한 것은 아니었다. 오히려 웹 애플리케이션의 컴포넌트화를 통한 재사용성과 통합에 초점을 둔 것이며, 재사용성의 장점을 살려 비즈니스 모델을 적용하다 보니 전자거래와 연관성을 갖게 된 것이다.

웹 서비스는 공개된 표준 기반의 인터페이스 제공과 하드웨어, 운영체제 및 프로그램 독립성 지향을 기본 전략으로 하고 W3C, OASIS, WS-I 등 웹 서비스와 관련된 국제 표준화 기구를 중심으로 활동이 이루어지고 있다.

본 고에서는 지금까지 다양한 매체를 통해서 다루어졌던 ebXML에 대한 내용보다는 동적 비즈니스 실현을 예고하는 웹 서비스에 대한 개념과 요소기술들, 그리고 문제점 및 전망에 대해서 살펴보도록 하겠다.

웹 서비스란?

웹 서비스란 용어에 대해 많은 사람들은 기존의 웹 페이지를 통한 정보제공 내지는 이를 위한 웹 호스팅 서비스 정도로 이해하고 있는 사람들이 많다.

그러나 최근 회자되고 있는 웹 서비스란, 사내 또는 외부에서 인터넷을 통해 제공되는 컴포넌트(Component)들을 애플리케이션-대-애플리케이션(application-to-application)간에 자동으로 연결시켜 가상의 애플리케이션을 구축할 수 있도록 하는 것으로 이상적으로는 전 세계 컴퓨터를 하나로 통합하기 위한 움직임이라 할 수 있다.

블록완구에 비유할 수 있는 컴포넌트란 'Method', 'Software function', 'Application' 단위로 이루어질 수 있는데, 제공자는 최소한 'Method' 단위로 먼저 구성하고, 이들을 확장시켜 'Software function', 'Application' 단위로 제공할 필요가 있다. 왜냐하면, 제공자가 서비스하는 임의의 단위를 이용자 모두가 만족할 수는 없기 때문에 이러한 경우는 이용자가 필요한 기능 단위로 이용할 수 있도록 해야 한다.

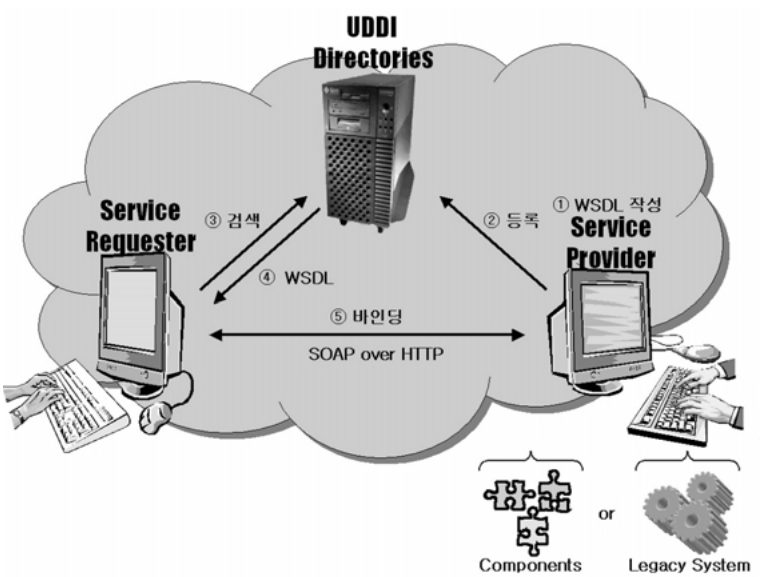
웹 서비스가 구현되면 시간과 장소의 제약을 받지 않고, 개발 언어 및 플랫폼에 관계없이 정보교환과 애플리케이션간의 연동을 통해 협업이 가능해져 진정한 의미의 분산 컴퓨팅 환경구현과 동적인 비즈니스 웹 환경구축의 실현을 가속화할 것으로 예견된다.

웹 서비스를 위한 요소기술

웹 서비스는 Publication/Discovery, Description, Messaging Transport network로 구성되는 기본계층과 Transaction, QoS, Service Context/Privacy, Reliable Messaging, 그리고 Business Workflow로 구성된 확장계층으로 구분될 수 있다. 기본계층은 단순 요청-응답에 기반한 메시지 교환 서비스를 제공하고 있으나, 확장계층은 단순 요청-응답 유형뿐만 아니라 협업 비즈니스 프로세스의 처리까지 지원하도록 하고 있다.

그러면, 웹 서비스 시나리오를 통해 어떠한 기술들이 사용되어 서비스를 이용할 수 있게 하는지 살펴보도록 하자.

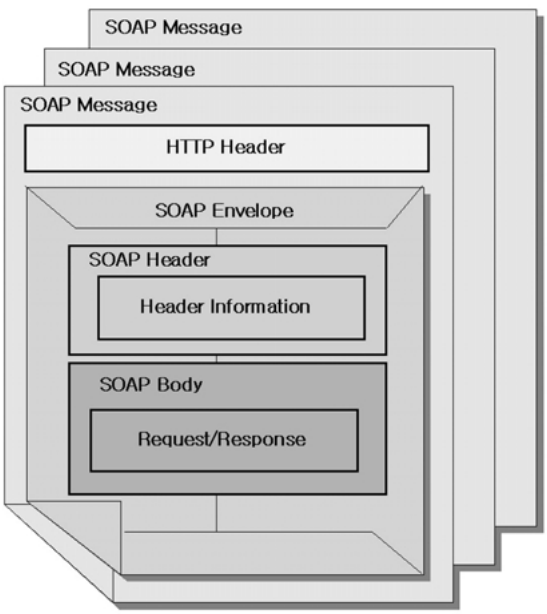
먼저, 서비스 제공자인 기업은 자신의 비즈니스 정보 및 제공할 서비스 정보, 그리고 이를 이용할 수 있는 방법 등에 대한 정보를 WSDL 파일에 작성하고, 이를 UDDI 레지스트리에 등록해 놓는다. 이때 등록되는 비즈니스 개체나 서비스는 UDDI에 기반한 고유 ID를 부여받게 된다. 그러면, 특정 서비스를 받고자 하는 서비스 요청자는 요구되는 서비스에 대한 메시지를 작성한 후, UDDI 레지스트리에서 분류 및 식별체계와 같은 여러 검색조건을 통해 검색하여 찾게 된다. 다음으로 해당 서비스를 이용하기 위한 WSDL 파일을 받게 되고, 이 정보를 이용해 서비스 제공자와의 연결을 통한 서비스를 제공받게 되는데 그 실행결과 값만을 XML 문서형태로 받게 된다.



[그림 1] 웹 서비스 시나리오

메시지 전송을 위한 'SOAP'

SOAP이란, 분산환경에서 소프트웨어 서비스들간에 정보를 교환하기 위한 XML 기반의 간단한 프로토콜로 메시지 내용과 이를 처리하는 방법을 설명하기 위해 프레임워크를 정의한 Envelope, 응용프로그램에서 정의한 데이터 타입의 인스턴스를 나타내는 일련의 인코딩 규칙, 원격 프로시저 호출 및 응답 등을 나타내는 규칙으로 구성되어 있다.



[그림 2] SOAP 메시지 구조

SOAP 메시지는 루트 엘리먼트로 Envelope를 가지며, SOAP Header와 Body를 하위 엘리먼트로 가지고 있다.

XML 포맷으로 표현된 SOAP 메시지는 전송계층(Transport Layer)을 통해 전송되어지는데 일반적으로 HTTP가 이용된다. 이외에도 SMTP(Simple Mail Transport Protocol), FTP(File Transfer Protocol) 등과의 바인딩을 통해 전송될 수도 있다.

SOAP Header 블록은 메시지 내에 없을 수도 있지만 존재하게 될 경우에는 SOAP Envelope 내의 처음에 위치해야 하며, 메시지가 가지는 우선 순위와 메시지의 유효기간에 대한 정보를 담고 있어야 한다.

Body 블록은 SOAP 메시지 내에 반드시 존재해야 하는 엘리먼트이며, 전송될 메시지의 내용을 담고 있다. 또한, 수신된 메시지를 애플리케이션이 잘 처리할 수 있도록 적합한 네임 스페이스를 반드시 가져야 하며, 에러 메시지를 지정하는 Fault 엘리먼트를 정의할 수 있다.

클라이언트는 먼저 웹 서비스를 기술한 WSDL 파일을 읽고 나서 웹 서비스를 위해 위에서 설명된 메시지 구조를 가지는 SOAP 메시지를 보내기 시작한다.

웹 서비스 기술(Description)을 위한 'WSDL'

WSDL(Web Service Description Language)은 원하는 서비스가 어디(Where)에 존재하며, 웹 서비스로 무엇(What)을 할 수 있고, 또 이를 실행하기 위해서는 어떻게(How) 해야 하는가를 XML 형식으로 제공되는 메타 언어라 할 수 있다. 이해를 돕기 위해 굳이 비유하자면 제품의 사용설명서와 같다고 할 수 있다.

이러한 WSDL은 여러분들이 잘 알고 있는 CORBA나 COM의 IDL(Interface Definition Language)이나 타입 라이브러리와 유사하다.

WSDL 문서는 'definitions'를 루트 엘리먼트로 가지며 크게 Type, Message, PortType, Binding, Port, Service 등 6가지의 엘리먼트로 구성되어 있다.

Types 엘리먼트는 메시지에 사용되는 데이터 타입(int, String, Object 등)의 정의를 기술하는데 사용되

는데, XSD(XML Schema Definition)를 기본 타입 시스템으로 사용한다.

Message 엘리먼트는 하나 이상의 논리적인 part들로 구성되어 있는데, 각 part는 특정 타입 시스템의 정의와 연관되어 있으며, 이와의 연결을 위해서 element와 type이라는 두 개의 애트리뷰트를 가지고 있다.

PortType 엘리먼트는 operation의 집합으로 구성되는데, 각 operation의 입출력에 사용되는 메시지는 input과 output을 통해 정의된다. 그리고 PortType의 name 애트리뷰트는 고유한 이름이어야 하며, 이를 통해 모든 operation들이 정의된다.

binding 엘리먼트는 특정 PortType에서 정의된 operation과 input/output message에 대한 프로토콜과 데이터 포맷 정보를 정의하는데 사용되며, 특정 PortType에 대한 바인딩 수는 제한이 없다.

Port 엘리먼트는 binding에 대해 하나의 주소를 지정하여 endpoint를 정의하는데 사용된다. name 애트리뷰트는 WSDL 문서 내에 정의된 포트들 중에서 유일한 Port 이름을 정의하는데 사용되며, Qname 타입의 binding 애트리뷰트는 binding을 참조하는데 사용된다. 그리고, 바인딩 확장 엘리먼트는 포트에 대한 주소정보를 정의하는데 사용된다.

하나의 Service 엘리먼트는 동시에 여러 Port들을 통해서 접근이 가능하다. 서비스 엘리먼트의 name 애트리뷰트 역시 WSDL 문서 내의 서비스들 중에서 유일한 이름을 가져야 한다.

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
    <import namespace="uri" location="uri"/>*
    <wsdl:documentation .... /> ?
    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xsd:schema .... />*
        <-- extensibility element --> *
    </wsdl:types>
    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>
    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
          <wsdl:documentation .... /> ?
          <wsdl:input name="nmtoken"? message="qname">?
              <wsdl:documentation .... /> ?
          </wsdl:input>
          <wsdl:output name="nmtoken"? message="qname">?
              <wsdl:documentation .... /> ?
          </wsdl:output>
          <wsdl:fault name="nmtoken" message="qname"> *
              <wsdl:documentation .... /> ?
          </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation .... />?
        <-- extensibility element --> *
        <wsdl:operation name="nmtoken">*
          <wsdl:documentation .... /> ?
          <-- extensibility element --> *
          <wsdl:input> ?
              <wsdl:documentation .... /> ?
              <-- extensibility element -->
          </wsdl:input>
          <wsdl:output> ?
              <wsdl:documentation .... /> ?
              <-- extensibility element --> *
          </wsdl:output>
          <wsdl:fault name="nmtoken"> *
              <wsdl:documentation .... /> ?
              <-- extensibility element --> *
          </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="nmtoken"> *
        <wsdl:documentation .... />?
        <wsdl:port name="nmtoken" binding="qname"> *
          <wsdl:documentation .... /> ?
          <-- extensibility element -->
        </wsdl:port>
        <-- extensibility element -->
    </wsdl:service>
    <-- extensibility element --> *
</wsdl:definitions>
```

[그림 3] WSDL 문법구조

웹 서비스 등록 및 검색을 위한 'UDDI'

UDDI는 비즈니스 정보와 서비스에 대한 정보를 전자적으로 발행하고 질의하기 위한 새로운 표준이다.

발행된 정보는 하나 또는 그 이상의 UDDI 레지스트리에 저장되며, 웹브라우저나 SOAP을 통해서 엑세스할 수 있다.

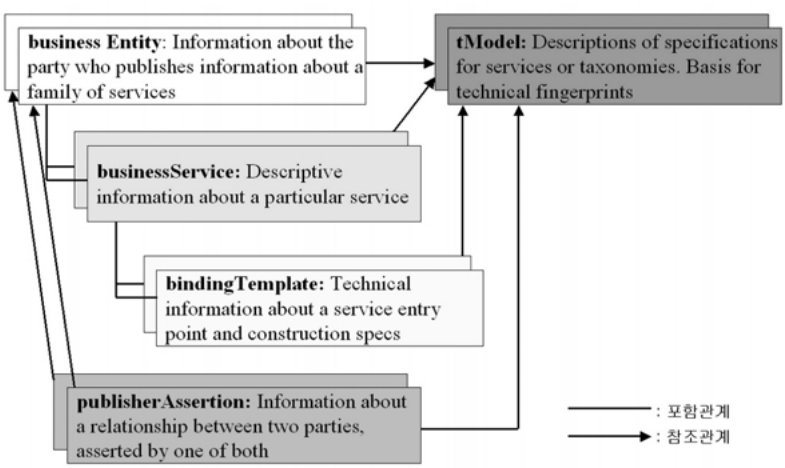
UDDI는 산업분류 코드 정보를 가지는 "Yellow pages"와 전화 및 팩스, 주소, 웹 사이트 등의 연락처와 위치, 식별자 정보를 가지는 "White pages", 그리고 비즈니스 서비스를 이용하는 방법에 대한 기술적 정보를 포함한 "Green pages"의 구조를 가지고 있다. 이러한 UDDI 레지스트리는 기업, 국가 또는 글로벌 단위로 운영될 수 있으며, 각 레지스트리간에 계층적인 관리체계를 가지고 있다. 레지스트리에 저장되는 서비스, 비즈니스 정보들의 정확성과 신뢰성 보장을 위해서는 체계적인 운영이 요구되며, 이를 위해 UDDI 버전 3.0에서는 계층적 관리체계를 규정하고 있다.

UDDI 레지스트리는 공용(Public) 레지스트리와 사설(Private) 레지스트리로 구분된다. 공용 레지스트리는 글로벌 환경에서의 서비스와 비즈니스에 대한 등록 정보를 제공하며 IBM, MS, SAP, HP와 같은 회사들이 운영하고 있으며, 현재 아시아 지역에서는 유일하게 NTT가 구축중에 있다. 사설 레지스트리는 기업 내 인트라넷 등 제약된 환경에서의 서비스와 비즈니스 정보를 제공한다.

사설 레지스트리는 Sun One Registry Server, IBM Private UDDI Registry, UDDI Server in Java 등이 존재하며, 기업 내부 또는 기업간 애플리케이션 통합에 사용된다.

국내의 경우, 아직까지 공용 레지스트리 구축사례는 없으며, SI 업체와 통신업체를 중심으로 사설 레지스트리 구축이 추진중에 있다. 동양시스템즈가 MS의 협력하에 .NET 기반의 K-UDDI 프로젝트를 추진중에 있다. 그리고 SK Telecom은 NATE 기반으로 Personal Platform 구축을 위해서 UDDI 파일럿 프로젝트를 계획하고 있으며, LG-CNS는 내·외부에서 제공하는 웹 서비스중 사용가능한 웹 서비스 정보를 등록, 검색하기 위한 웹 서비스 카탈로그 용도의 사설 레지스트리를 구축하여 운영중에 있다.

UDDI 버전 2.0의 데이터 구조는 businessEntity, businessService, bindingTemplate, tModel, PublisherAssertion을 포함한 5가지 구성요소를 가지고 있다.



[그림 4] UDDI 데이터 구조

'웹 서비스를 위한 보안' WS-Security

웹 서비스 분야에서 가장 많이 이야기가 되고 있는 것이 웹 서비스 보안인데, 웹 서비스의 확산 및 구현속도를 더디게 하는 요소 중의 하나를 꼽으라면 바로 웹 서비스 보안 표준의 부재가 아닌가 싶다. 이에 많은 업체들은 웹 서비스 구현에 신중을 기하고 있다.

최근 고객과 기업의 웹 서비스에 대한 요구에 의하여 IBM, Microsoft, 그리고 VeriSign이 새로운 웹 서비스 보안 명세서인 WS-Security를 발표했다. 표준으로 제안된 WS-Security 명세서는 포괄적인 웹 서비스 보안 모델을 제시하고 있는데, 안전한 웹 서비스를 구축하도록 하는데 목적이 있다.

웹 서비스 표준화 계획에 근거하여 개발된 웹 서비스 보안명세들에 대한 로드맵은 웹 서비스 환경에서의 메시지 교환에 대한 보안을 어떠한 방법으로 제공할 것인지를 정의하고 있으며, http://www-106.ibm.com/developerworks/security/library/ws-secmap에서 직접 확인할 수 있다.

본 로드맵은 WS-Security 명세서에 의해 수립된 프레임워크 안에 기본적인 보안 접근법 및 부가적인 정의, 그리고 웹 서비스 보안능력 등에 근간을 두고 있다. 크게는 WS-Policy, WS-Trust, WS-Secure Conversation, WS-Federation, WS-Authorization을 포함하고 있다.

WS-Security는 SOAP 확장 표준으로 정의할 수 있는데, 메시지를 암호화시켜 안전성 있게 교환할 수 있는 일반적인 보안모델, 메커니즘, 그리고 기술들의 통합을 지원하여 다양한 보안 토큰을 적용할 수 있도록 하고 있다. 이 WS-Security는 SOAP 메시징에 "무결성(Integrity)"과 "기밀성(Confidentiality)", 그리고 "인증(Authentication)" 사항을 추가하여 보안 기능을 강화한 것이다.

메시지가 수정없이 전달되었다는 것을 보증하기 위하여 보안 토큰에 XML Signature를 사용하는데, 이 무결성 메커니즘은 다양한 서명방법과 'Actor', 그리고 추가적인 서명 포맷을 지원하기 위한 확장성을 가지도록 디자인되었다.

또한, SOAP 메시지가 기밀성을 유지하도록 하기 위해 보안 토큰에 XML Encryption을 사용한다. 암

- **WS-Policy** : 송수신자간의 보안 요구사항과 프라이버시, 인코딩 포맷, 지원 알고리즘을 포함하는 여러 가지 기본 서비스 애트리뷰트들을 명시하는 보안정책 사항과 제약을 표현하기 위한 방법을 정의하고 있다.
- **WS-Trust** : "직접"과 "중개"를 거치는 신용관계의 유지 및 해지를 설정하기 위한 모델을 기술하고 있으며, 기존의 보안 토큰 발행 서비스가 WS-Trust와 함께 사용될 수 있는 방법 또한 제공하고 있다.
- **WS-Privacy** : WS-Security, WS-Policy, WS-Privacy를 함께 사용하여 기업들은 기술된 프라이버시 정책에 대한 적합성을 기술하고 지적할 수 있다.
- **WS-Secure Conversation** : 메시지 교환 보안이 어떤 방식으로 관리되는지를 명시하는 것으로써 웹 서비스가 요청자의 메시지를 입증하는 방법과 요청자가 서비스를 입증하는 방법, 그리고 상호 입증된 보안내용들을 입증하는 방법들에 대해서 기술하고 있다. 또한, 시스템 보안을 보다 더 강화하기 위해 전송계층 보안은 WS-Security와 WS-Secure Conversation을 함께 사용할 수 있다.
- **WS-Federation** : 이질적인 분산환경에서의 신용관계(trust relationship)를 유지하고 해지할 수 있는 관리 메커니즘을 포함하고 있다.
- **WS-Authorization** : 웹 서비스 접근정책의 명시와 관리방법을 기술하고 있으며, 특히 claim이 보안 토큰 내에 명시되는 방법과 이러한 claim들이 종단에서 해석되는 방법을 포함하고 있다. 또한, 인증 포맷과 언어에 대한 유연성과 확장성을 갖도록 설계되어 있다.

호화 메커니즘은 추가적인 암호화 기술, 프로세스, 다양한 'Actor'에 의한 오퍼레이션을 지원하도록 디자인되었다.

앞서 살펴본 바와 같이 WS-Security는 보다 안전하고 상호운용 가능한 웹 서비스를 구축 · 개발하기 위해 중요한 기본 레이어를 제공한다. WS-Security의 다른 중요 요소는 견고성과 개방형 표준 기반의 보안 모델, 그리고 빠른 개발이다.

이 웹 서비스 보안 모델은 기존의 Transport Security, PKI, Kerberos 등과 같이 데이터 무결성과 기밀성에 대한 보안 모델들과 호환이 가능하여 결과적으로 웹 서비스 기반의 솔루션들과 통합이 가능하다. 이러한 통합이 갖는 의미는 애플리케이션 보안의 기능적인 요구사항들과 특정 메커니즘을 분리시켜야 함을 내포하고 있다. 예를 들어, 온라인으로 구매한 고객이 셀룰러 폰을 사용하든 랩탑 컴퓨터를 사용하든, 각각의 디바이스가 안전하고 올바르게 신원을 확인하기만 한다면 어떠한 요소에도 영향을 받지 않아야 한다는 것이다.

보안 모델들은 모두 호환이 가능하지만 상호운용성 확보를 위해 Signature와 Encryption을 위한 어댑터와 공통 알고리즘이 개발되거나 그에 대한 합의가 필요하다 하겠다.

도전 과제와 다양한 시도들

웹 서비스가 IT 기업들에게 새롭고 지속적인 매출을 보장해 줌과 동시에 일반 기업과 사용자들에게 비용절감과 새로운 사용자 경험을 제공해주기 위해서는 상호운용적인 웹 서비스 표준에 대한 동의가 전제되어야 한다. SOAP, UDDI, WSDL과 같은 기본적인 표준에 대한 동의는 명백하나 웹 서비스의 확장성을 고려한 QoS(Quality of Service), 권한부여(Authorization), 트랜잭션(Transaction) 보안 등에 대한 문제들에 대해서는 합의가 이루어지지 않고 있다. 특히, 가장 큰 문제점으로 지적되고 있는 보안문제는 현재 주요 웹 서비스 관련 표준화기관과 애플리케이션 벤더들의 가장 큰 해결과제로, 관련 표준과 솔루션들이 개발중에 있다. 웹 서비스 보안문제가 해결되면 일반 기업과 B2B 전자거래에서의 웹 서비스 활용이 더욱 빠르게 확산될 수 있을 것으로 기대된다. 그러나, 각 벤더들마다 미해결 문제에 대해 나름대로의 표준들을 개발하고 있어 완벽한 상호운용성을 가진 웹 서비스 제공에는 한계가 있다.

웹 서비스에는 XML, WSDL, SOAP, UDDI와 같은 기본 표준 이외에 다양한 표준들이 있으며, 현재도 새로운 표준과 스펙들이 개발되고 있다.

새로운 웹 서비스 표준들은 웹 서비스 기본 표준을 보완하고 강화하기 위해 개발되고 있다. 이와 같은 새로운 표준을 통해 완성도 높은 웹 서비스를 구현하는 것이 가능하게 될 것이다. 하지만, 벤더와 표준화기관의 경쟁적 표준 개발은 오히려 웹 서비스 상호운용성에 부정적인 영향을 미칠 가능성이 있는 것도 사실이다. 현재 주목받고 있는 새로운 웹 서비스 표준으로는 다음과 같은 것들이 있다.

WSCI(Web Services Choreography Interface)

썬마이크로시스템즈, SAP, BEA, 인탈리오(Intalio) 등이 2002년 6월 발표한 웹 서비스 표준으로 기존의 WSDL 기능을 보완한 새로운 표준이다. WSCI는 웹 서비스간 메시지의 송/수신만을 정의하는 정적(Static)인 WSDL이 웹 서비스간 다양하고 복잡한 상호작용을 충분히 표현해 내지 못한다는 단점을 보완하는 동적(Dynamic)인 웹 서비스 기술표준이다.

WSCI는 WSDL을 포함한 ebXML의 BPSS나 MSFT의 XLANG, BTP, BPML, WSCL 등과 같은 다양한 웹 서비스 관련 표준(또는 스펙)과의 유연한 연동을 지원하면서 웹 서비스의 동적 인터페이스를 정의하는 역할을 한다.

BPEL4WS(Business Process Execution Language for Web Services)

BPEL4WS는 IBM의 WSFL(Web Services Flow Language)과 마이크로소프트의 XLang을 결합한 것으로 웹 서비스 환경에서 서로 다른 비즈니스 프로세스가 서로를 인식하고 이해하는 것을 가능케 해주는 XML 기반 플로우 언어(Flow Language)이다. BPEL4WS는 기업내 또는 기업간 비즈니스 프로세스 간의 상호작용과 통합을 지원한다.

WS-Coordination

WS-Coordination은 이름이 의미하는 것처럼 개발자가 웹 서비스 환경에서 특정 기능을 가지고 있는 웹 서비스들의 작업을 관리하고 조정(coordination)할 수 있도록 하는 스펙이다. WS-Coordination은 웹 서비스 환경에서 분산되어 있는 웹 서비스의 작업 수행을 조정하는 트랜잭션 프로토콜(Transaction Protocols)을 형성하고 등록하는 표준 메커니즘을 제공함으로써 웹 서비스간의 조정(Coordination)이 발생할 수 있는 구조(Structure)를 제공한다.

WS-Transaction

WS-Transaction은 비즈니스 프로세스에서 특정 기능의 성/패 여부를 모니터 할 수 있도록 하는 스펙이다. WS-Transaction은 웹 서비스 환경에서 분산되어 있는 조직간에 안정적인 거래를 지원할 수 있는 유연한 트랜잭션 프로토콜을 제공하며, 특정 기능의 오류를 감지하고 이에 대응할 수 있도록 해준다.

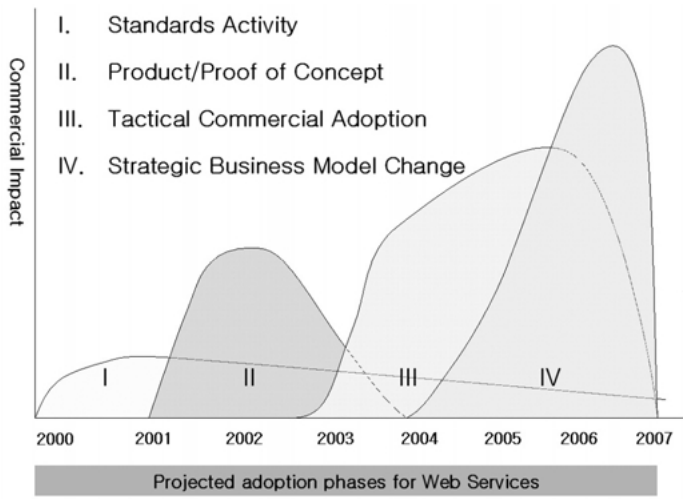
WS-Coordination과 WS-Transaction은 함께 사용됨으로써 웹 서비스의 활용도를 더욱 높게 해 줄 수 있다. 예를 들어 항공편 예약, 호텔 예약, 렌터카 예약과 같은 다양한 기능들이 포함되어 있는 "여행준비"라는 웹 서비스를 수행하는데 WS-Coordination과 WS-Transaction이 적용됨으로써 각 기능들이 해당되어 있는 서비스의 기술기반이나 제공회사에 관계없이 "여행준비"라는 전체 기능을 완료할 수 있도록 해준다.

시장 및 발전전망

이전에는 플랫폼 종속적인 환경이 지배적인 관계로 자바와 COM을 함께 사용하는 것은 어려운 일이었다. 실제로 자바 애플리케이션이나 COM 애플리케이션들은 각각의 애플리케이션들간에만 호환성을 지원해주었다. 그러나, 웹 서비스를 이용하는 경우에는 정보를 찾는 방법과 웹을 통해 그 정보를 입수하는 방법에 대한 것만 고려하면 된다. 이제까지 중요하게 여겨진 프로그래밍 언어, 운영체제, 객체 모델, 어느 벤더가 제공하는 데이터베이스인가 등에 대한 이슈들은 이제 고려할 필요가 없을 것이다.

웹 서비스는 기존의 기술 인프라구조를 대신하지는 않을 것이며, 기존에 존재하는 기술들을 통합하는데 사용되어질 것이다. 예를 들어, J2EE 애플리케이션과 COM 애플리케이션간의 메시지 교환이 필요한 경우 또는 함께 동작해야 하는 경우 통합 채널(Integration Channel)을 제공함으로써 쉽게 해결해 줄 것이다.

웹 서비스는 현재 소프트웨어 벤더와 IT 서비스 업체 그리고 기업 IT 부서의 최대 관심사이기는 하지만 도입과 대중화까지는 상당한 시간이 소요될 것으로 전



[그림 5] 웹 서비스 시장 발전단계(출처 : Hurwitz Group)

망된다. 지금까지 살펴본 것과 같이 웹 서비스 기술이 아직까지는 초기 단계(early stage)에 불과하고 다양한 표준(또는 스펙) 및 솔루션 개발을 통해 기술적, 기능적 완성도를 추구하는 단계이기 때문이다.

IT 리서치 기관인 허윗츠 그룹(Hurwitz Group)은 웹 서비스 시장 발전단계를 크게 4단계로 구분하였다[그림 5].

웹 서비스 개념과 관련 표준들이 개발되기 시작한 2000년부터를 표준화 단계(Standards Activity)로 명명하고 있으며, 2001년부터 2004년 사이에는 웹 서비스 관련 제품이 개발되고 웹 서비스에 대한 명확한 개념정의가 활발하게 진행될 것이라고 전망했다.

실제로 웹 서비스 도입이 시작되는 것은 2003년 전후로 웹 서비스 표준화가 어느 정도 진행되고 웹 서비스 관련 제품의 지원이 충분해진 뒤에나 가능할 것으로 전망하였다.

일단 상업적인 웹 서비스 도입이 시작되면 웹 서비스를 통한 기업의 비즈니스 모델과 비즈니스 프로세스 변화는 즉각적으로 시작될 것이다. 웹 서비스가 활성화되는 시점은 다수의 기업들이 웹 서비스를 도입해 웹 서비스를 통해 비즈니스 프로세스가 변화되는 시점인 2005년 전후가 될 것으로 전망된다.

또 다른 리서치 기관인 가트너그룹은 내부 애플리케이션에의 적용은 2003년부터 시작될 것이며, 외부 파트너와의 연계는 2004년부터, 이와 관련된 시장은 2003년 하반기부터 열릴 것으로 전망하고 있다. 또한, 2005년까지 웹 서비스 관련 솔루션 시장이 280억 불에 달할 것으로 전망하고 있다.

참고자료

[1] Eric Newcomer, Understanding Web Services, Addison-Wesley, 2002

[2] Graham Glass, Web Services, Prentice Hall, 2002

[3] S. Jeelani Basha, Scott Cable, Ben Galbraith, Mack Hendricks, Romin Irani, James Milbery, Tarak Modi, Andre Tost, Alex Toussaint, Professional Java Web

Services, Wrox Press Ltd., 2002
[4] Patrick Cauldwell, Rajesh Chawla, Vivek Chopra, Gary Damschen, Chris Dix, Tony Gong, Francis Norton, Uche Ogbuji, Glenn Olander, Mark A. Richman, Kristy Saunders, Zoran Zaev, Professional XML Web Services, Wrox Press Ltd., 2001
[5] Steve Graham, Simeon Simeonov, Toufic Boubez, Doug Davis, Glen Daniels, Yuichi Nakamura, Ryo Neyama, Bulding Web Services with Java, SAMS, 2002
[6] Simon St.Laurent, Joe Johnston, Edd Dumbill, Programming Web Services with XML-RPC, O' REILLY, 2001
[7] Broan A. LaMacchia, Sebastian Lange, Matthew Lyons, Rudi Martin, Kevin T. Price, .NET Framework Security, Addison-Wesley, 2002
[8] Blake Dournaee, XML Security, McGraw-Hill, 2002
[9] Anup K.Ghosh, Security & Privacy for E-Business, John Wiley & Sons, Inc., 2001
[10] http://www.w3.org/TR/SOAP, 8 May 2000
[11] http://www.w3.org/TR/wsdl, 15 March 2001
[12] http://www.uddi.org
[13] http://www.oasis-open.org/committees/uddi-spec
[14] http://www.ws-i.org
[15] http://www.webservices.org/

정통부, 상호접속기준 개정 위피 제도화

정보통신부는 국내 무선인터넷플랫폼 표준인 위피'(WIPI)를 무선인터넷 서비스 제공에 필요한 상호접속기준으로 채택하기 위해 전기통신설비 상호접속기준의 개정을 추진키로 했다고 지난 1월 6일 밝혔다. 이를 위해 정통부는 위피의 제도화방침을 세계무역기구(WTO)의 무역기술장벽(TBT)위원회에 이같은 방침을 통보, 60일간 WTO 회원국으로부터 의견을 수렴한 후 상호접속기준의 개정 작업에 들어갈 계획이다. 이와 함께 위피 제도화를 심층적으로 논의하기 위해 한 · 미 양국의 민 · 관 기술전문가가 참여한 토론회를 개최해 양국간 견해 차이를 조율하기로 했다. 상호접속기준은 무선인터넷 서비스를 위해 이동통신 사업자들이 반드시 준수해야 하기 때문에 위피가 상호접속기준으로 채택되면 사업자들은 이를 의무적으로 이용해야 한다. 현재 국내에서는 이동통신 사업자별로 5개의 독자규격 플랫폼을 사용하고 있어 콘텐츠 제공업체(CP)의 경우 사업자에 맞게 별도의 콘텐츠를 제작해야 하는 상황이다. 정통부는 위피 사용을 의무화함으로써 무선인터넷 업계의 콘텐츠 개발 부담을 줄이는 동시에 무선인터넷 시장이 공정경쟁 환경과 소비자 선택권 확대 효과를 거둘 것으로 기대하고 있다.