웹서비스 기술과 응용사례

김성익* · 오찬주** · 최영미***

1. 서 론

사용자는 때와 장소에 구애받지 않고 인터넷을 통해 다양한 동적인 정보를 실생활 및 비즈니스에 이용하기를 원하고 있다. 이에 부응하여 산업체에서는 기존의 시스템을 e-비즈니스화 하면서 다양한 분야에 개방과 통합의 작업을 하고 있다. 이같이 산업체에서 주도하고 있는 재통합 움직임으로 대표되는 것이 웹서비스이다. 웹서비스는 전통적인 프로그래밍 방식인 컴파일을 통한 정적 바인딩이라는 이슈에서 실행할 때 동적으로 서비스를 검색하여 처리하는 살아 움직이는 e-비지니스의 서막을 알린다는 면에서 그 의미가 깊다.

웹서비스가 추구하는 것은 특정 벤더 기술에서 벗어나 표준을 통한 모든 컴퓨터 간의 원활한 데이터의 흐름이다. 웹서비스는 하드웨어 및 소프트웨어의 장벽을 넘어 사용자가 정보를 손쉽게 이용할 수 있도록 플랫폼 환경을 구축하는 아키텍처로 SUN, MS, IBM, 오라클, BEA 등이 주도적으로 구현에 나서고 있다. 본 연구에서는 웹서비스의 기본 기술을 살펴본 후 응용사례로 방카슈랑스 시스템과 웹서비스 구현 플랫폼 WebInOne[2]을 소개하면서 현장에서 핵심기술이 적용되는 방식을 보이고자 한다.

* TmaxSoft 책임컨설턴트
** 동양시스템즈 기술연구소 책임연구원
*** 성결대학교 멀티미디어학부 부교수

본 논문은 다음과 같은 내용으로 구성된다. 2장에서는 웹서비스의 개요로 정의와 특징에 대하여 기술한다. 3장에서는 웹서비스 관련 구현기술로 아키텍처와 핵심기술에 대하여 설명한다. 4장과 5장에서는 웹서비스의 구현사례로 방카슈랑스와 웹서비스 플랫폼을 소개한다. 6장에서는 결론과 앞으로의 연구방향을 기술한다.

2. 웹서비스 개요

2.1 웹서비스 정의

• 웹서비스 정의1

웹 서비스는 표준 기술(XML, HTTP, RPC)에 기초하여 이질적인 환경에 있는 프로그램 간에 통신할 수 있게 하는 기술이다.

• 웹서비스 정의2

웹 서비스란 이미 존재하는 구현 내용이나 새로운 구현 내용을 다양한 업계 표준 기술을 사용하여 웹을 통하여 사용자에게 또는 다른 서비스에게 제공하기 위한 오픈 스탠더드이다.

• 웹서비스 정의3

A Web services is a collection of functions that are packaged as a single entity and published to the network for use by other programs.

이상의 정의를 종합하면, 웹서비스는 "다른 프로그램들이 표준 XML과 인터넷 프로토콜을 이용하여 네트워크 상에서 사용할 수 있는 함수들 또는 연산들의 모임"이라고 할 수 있으며, 분산객체기술(Distributed Object Technologies)과 웹(Web)의 장점을 결합한 형태라고 볼 수 있다[12,15].

따라서 CORBA(Common Object Request Broker Architecture), DCOM(Distributed Component Object Model) 등과 같은 분산객체 기술이 제공하는 네트워크에 분산된 객체들에 대한 투명한 접근성을 보장한다. 또한 HTTP와 XML과 같은 표준프로토콜과 데이터 형태(Data Format)를 통하여 웹 서비스를 접근함으로써 DCOM RPC (Remote Procedure Call), IIOP(Internet Inter ORB Protocol), RMI(Remote Method Invocation)과 같은 분산객체 모델에 종속된 프로토콜을 사용하여 발생한 많은 문제점을 제공한다[7].

2.2 웹서비스 특징

웹서비스는 소프트웨어 기술에서는 분산객체 기술의 뒤를 잇는 계보를 가진다고 볼 수 있는데, 인터넷을 대상으로 하는 만큼 분산의 정도가 매우 크다는 특징이 있다. 이렇게 웹 서비스에 기반을 둔 프로그래밍 파라다임을 '서비스지향적인 컴퓨팅'이라고 한다. 서비스 지향적인 컴퓨팅은 기존의 프로그래밍을 높은 수준의 분산 애플리케이션으로 이끄는 새로운 패러다임이다. 웹 서비스는 그 자체로 실행가능한 모듈 형태를 띠고 있으면서 필요한 곳에 배포되거나 웹서비스 자체의 위치를 변경할 수 있다. 또한 웹의 어디에 있든지 실행할 수 있다는 특징이 있다[3,5,6].

3. 웹서비스 기술

3.1 웹서비스 아키텍쳐

이종 기기간의 통합을 위한 다방면의 노력은 국제적인 표준을 요구하게 되었고, 이를 위해서 XML 기술을 기반으로 하는 다양한 표준들(SOAP, XSD, WSDL, UDDI)의 기반 위에 XML 웹서비스가 만들어지게 되었다. 웹서비스는 웹환경에서의 시스템 간 통합을 위한 서비스 지향 아키텍쳐(SOA)의 핵심적 구성요소이다. 웹서버를 통해 제공되는 서비스를 제공하는 서비스 제공자와 이를 소비하는 서비스 소비자가 양자 모두 XML에 기반한 표준적인프로토콜을 dldydgkadmfhTJ 서로 다른 기반의 환경에서도 서비스를 통한 통합이 가능하게 되었다. 더불어 다양한 프로그래밍 프레임웍에서 이러한 XML 웹서비스의 구현은 매우 용이하고, 정확하게 사용이 되도록 지원되고 있다.

웹서비스 아키텍쳐는 아래 그림 1과 같이 서비스 제공자(Service Provider), 서비스 요청자(Service Requester) 및 서비스 중개자(Service Broker)와 같은 세가지 역할들의 상호작용을 기반으로 한다.

서비스 제공자는 비즈니스 관점에서 보면, 서비스 소유자이며, 아키텍쳐 관점에서는 서비스가 운영되는 플랫폼이다. 서비스 요청자는 비즈니스 관점에서 특정 서비스를 요구하는 비즈니스이며, 아키텍쳐 관점에서는 서비스를 찾고, 호출하는 어플리케이션이다. 서비스 중개자는 서비스 제공자

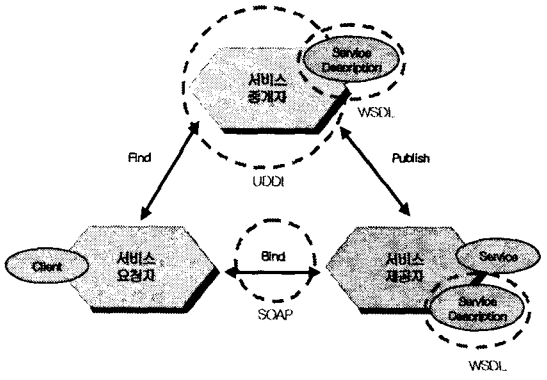


그림 1. 웹서비스 아키텍쳐

가 출판하는 서비스 기술(Service Description)들을 관리하고, 서비스 요청자에게 서비스 검색 서비스를 제공한다[9].

3.2 웹서비스 핵심기술

웹서비스를 구현하거나 웹서비스를 사용하기 위해서 필요한 핵심기술은 아래와 같다[4,6].

• XML(eXtensible Markup Language)

W3Cdml 텍스트 기반 마크업 언어에 대한 표준안으로 태그를 이용해서 각종 표현과 데이터를 나타낼 수 있다. XML은 이식성이 높은 구조화된 데이터를 기술하는 데 유용하며, 주로 데이터를 기술할 때 많이 이용한다. 특히 메시징 프로토콜과 상호교환을 필요로 하는 데이터 포맷으로 많이 이용하고 있다[17].

• HTTP(Hyper Text Transfer Protocol)

웹 서비스의 전송 네트워크로서 웹 서비스에 사용되는 XML 메시지를 전달하는 역할을 한다. HTTP와 같은 표준 전송 네트워크를 사용함으로써 다양한 플랫폼 간에 통신이 가능하다.

• SOAP(Simple Object Access Protocol)

분산 환경에서 정보를 교환하기 위한 목적으로 고안된 XML 기반의 경량 프로토콜로, 객체의 수요자와 제공자 사이의 메시징 프로토콜을 정의한다[13].

• WSDL(Web Services Description Language)

서비스 제공자가 보유하고 있는 서비스의 인터페이스를 XML을 사용하여 서비스 사용자들에게 제공하기 위한 웹 서비스 표준 기술(Description) 언어이다. 서비스 제공자는 WSDL을 통해 원격 메소드 호출(RMI)을 지원하기 위한 요청과 응답 메시지 포맷을 기술할 수 있다. 일반적으로 WSDL은 관련 연산을 정의하며, 매개변수와 데이터타입도 정의한다. 또한 서비스의 위치와 바인딩에 대한 세부 내용도 정의한다[16].

• UDDI(Universal Description discovery and Integration)

UDDI의 표준은 서비스 중개자를 구현하기 위한 SOAP API의 공통적인 집합을 제공한다. UDDI를 통하여 표준화된 방법으로 원하는 서비스를 검색하고 등록할 수 있다. IBM과 마이크로소프트, 아바라가 주도가 되어 웹 기반 서비스의 생성, 기술, 검색, 통합이 쉽도록 작성하였으며 uddi.org를 통해 표준화가 진행되고 있다[14].

다음 그림 2는 개념적인 웹서비스의 스택(Stack)을 보여 준다.

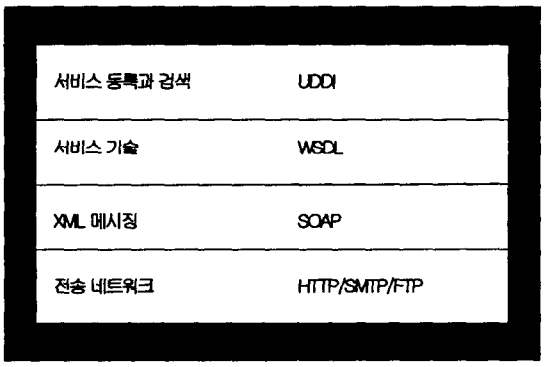


그림 2. 개념적 웹 서비스의 스택

4. 사례1(방카슈랑스 시스템)

4.1 방카슈랑스 개요

방카슈랑스(Bancassurance) 란 프랑스어로 방크(Banque, 은행)와 아슈랑스(Assurance, 보험)의 합성어로 기존 은행과 보험회사가 서로 연결하여 일반 개인에게 광역의 금융서비스를 제공하는 것이다[1].

우리나라는 국내 금융기관의 종합금융기관화를 촉진하고 금융산업의 국제 경쟁력을 키우기

위해 2003년 8월부터 은행, 증권, 상호저축은행 등 판매망을 갖춘 모든 금융기관(보험과 유사한 공제상품을 판매하는 농·수협, 신협, 우체국 등 제외)에 대해 대리점(중개사) 자격으로 보험상품을 판매할 수 있도록 허용하였다.

1단계로 8월부터 연금저축, 교육, 개인연금, 주택화재 등의 보험상품 판매를 허용하며, 2005년 4월 이후 암·종신보험 등 보장성보험, 자동차보험(개인용)을, 2007년 4월 이후엔 자동차(영업·업무용), 퇴직보험 등 모든 보험상품을 판매할 수 있다.

이에 따라 은행을 비롯한 모든 금융회사들은 보험사들과 전략적 제휴를 통한 보험상품 확보와 더불어 보험상품 판매를 위한 정보시스템의 구축이 필요하였다.

4.2 방카슈랑스 시스템의 개념적 아키텍쳐

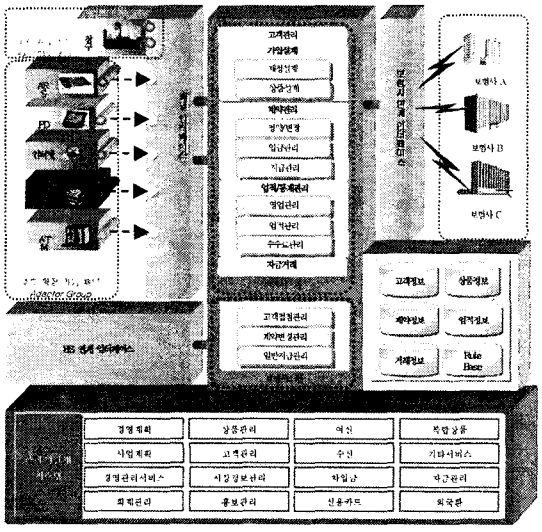


그림 3. 방카슈랑스시스템의 개념적 아키텍쳐

방카슈랑스 시스템은 위의 그림 3과 같이 고객관리, 가입설계, 업적/통계관리, 자금관리와 같은 연계시스템과 고객접점관리, 계약변경관리, 일반지급관리와 같은 보험시스템을 구축하는 것이다.

또한 다양한 고객 접근 채널을 위한 인터페이스, 기존 기업정보시스템과의 인터페이스와 제휴보험사들과의 인터페이스를 포함한다.

4.3 방카슈랑스의 웹서비스 적용사례

방카슈랑스 시스템에서 웹서비스는 기업정보시스템과의 인터페이스와 제휴보험사 인터페이스에 적용될 수 있다.

그림 4에서 웹 서비스를 기업내 통합에 적용하는 것은 웹 서비스 발전 단계에서 1단계에 해당된다. 이 경우에는 비즈니스 프로세스가 고정되어 있고, 참여자들간에 매우 잘 인식하고 있기 때문에 UDDI가 필요없다. 또한 비즈니스 통합(B2Bi)에 웹 서비스를 적용하는 것은 2단계에 해당된다. 이 경우에는 비즈니스 프로세스가 사적 표준화되어 있고, 참여자들간에 잘 인식하고 있기 때문에 역시 UDDI가 필요하지 않다.

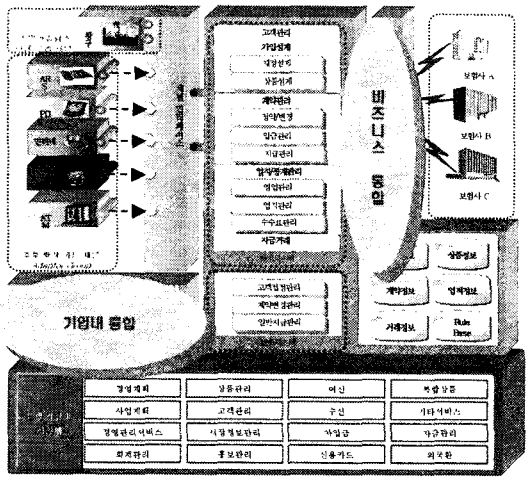


그림 4. 방카슈랑스시스템의 단계별 통합

4.4 웹서비스를 이용한 EIS 인터페이스 방법

웹 서비스를 기업내 통합에 적용한다는 것은

표준화된 인터페이스를 통하여 기업내에 존재하는 다양한 정보시스템에 접근할 수 있는 기반구조를 제공한다는데 의미가 있다.

아래 그림 5와 같이 방카슈랑스 시스템과 기업정보시스템은 SOAP 프로토콜을 이용하여 상호연동한다. 방카슈랑스 시스템은 기업정보시스템이 제공하는 WSDL로 정의되어진 표준 인터페이스를 통하여 비즈니스 로직 또는 데이터에 접근할 수 있다.

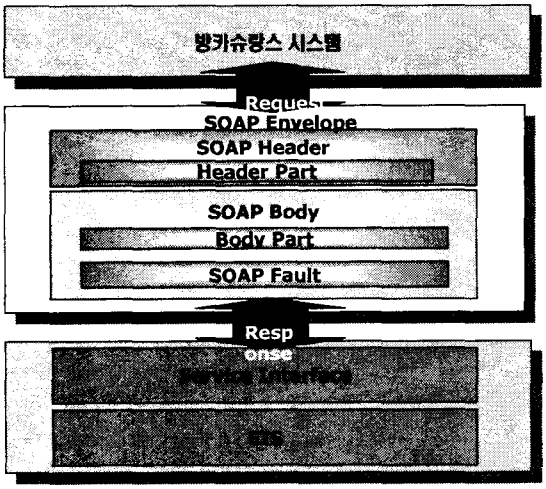


그림 5. SOAP프로토콜을 이용한 상호연동

4.5 웹서비스를 이용한 은행/보험사간 인터페이스

제휴관계를 형성한 보험사와 금융기관들의 인터페이스는 다-대-다 관계를 가진다. 은행 또는 보험사가 n개의 제휴관계를 가진다면 최대 n개의 서로 다른 인터페이스 시스템을 구축하여야 한다. 그러나 웹서비스를 적용할 경우에는 은행 또는 보험사는 단 하나의 인터페이스 시스템을 구축하면 된다. 아래 그림 6은 J2EE 기반으로 구축한 방카슈랑스 시스템의 응용 아키텍처이다.

은행시스템의 비즈니스 로직은 SOAP 프로토콜을 이용하여 보험사측의 웹서비스를 호출한다. 따라서 보험사는 제휴 금융기관이 추가될 경우 해당 웹서비스의 WSDL을 제공하고 은행은 이 WSDL을 이용하여 웹서비스 클라이언트를 개발함으로써 매우 빠르고 쉽게 인터페이스할 수 있다.

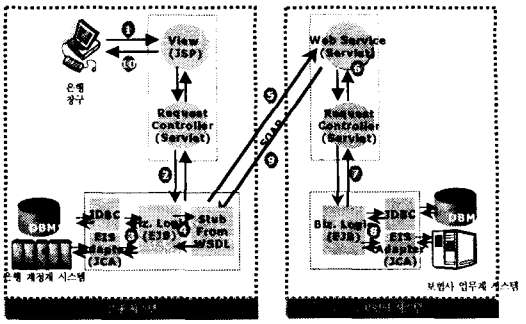


그림 6. J2EE 기반 방카슈랑스시스템 아키텍처

그림 7에서 .NET의 경우에도 동일한 방법으로 웹서비스 또는 웹서비스 클라이언트를 개발할 수 있다.

또한 웹서비스는 표준기술이므로 J2EE 기반으로 구축되어진 은행시스템과 .NET 기반으로 구축되어진 보험사 시스템간에도 적용이 가능하다. .NET 프레임워크 기반으로 웹서비스 클라이언트의 개발과정은 다음과 같다. 보험사와 제휴 금융기관이 주고받을 메시지를 아래와 같이 XML 문서로 정의하고, banka.xml로 저장하였다고 가정하자.

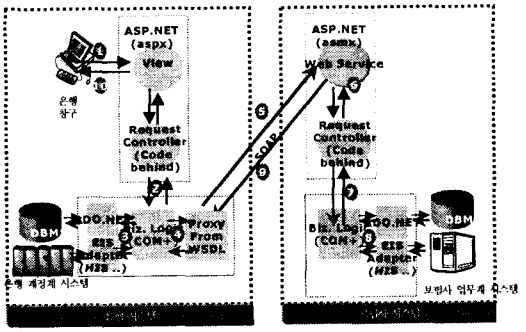


그림 7. .NET 기반 방카슈랑스시스템 아키텍처

```
<?xml version="1.0" encoding="EUC-KR"?>
        <Bancassurance>
                <HeaderArea>
                        <HDR_TRX_ID/>
                        <HDR_SYS_ID/>
                        ...
                </HeaderArea>
                <BusinessArea>
                        <PITEM/>
                        <PITEM_NM/>
                        ...
                </BusinessArea>
        </Bancassurance>
```

Visual Studio.NET 명령 프롬프트에서 xsd banka.xml을 실행하면 banka.xsd가 생성된다. 객체정보를 이용하여 직렬화(Serialization)를 하기 위해서 스키마 파일에서 클래스 파일을 생성하여야 한다. 명령 프롬프트에서 xsd banka.xsd를 실행하면 아래와 같은 cs파일이 생성된다.

```
using System.Xml.Serialization;
        /// <remarks/>
        [System.Xml.Serialization.XmlRootAttribute("B
        ancassurance", Namespace="", IsNullable=false)]
        public class Bancassurance {
          /// <remarks/>
          public              BancassuranceHeaderArea
        HeaderArea;
          /// <remarks/>
         public             BancassuranceBusinessArea
         BusinessArea;
         }
          /// <remarks/>
         public class BancassuranceHeaderArea {
            /// <remarks/>
            public string HDR_TRX_ID;
            /// <remarks/>
            public string HDR_SYS_ID;
            ...
         }
          /// <remarks/>
         public class BancassuranceBusinessArea {
            /// <remarks/>
            public string PITEM;
            /// <remarks/>
            public string PITEM_NM;
            ....
         }
```

이 생성된 객체를 이용하여 아래와 같이 웹서비스 클라이언트를 개발할 수 있다.

```
Bancassurance bs = new Bancassurance();
bs.HeaderArea = new BancassuranceHeaderArea();
bs.BusinessArea = new BancassuranceBusinessArea();
// 각 아이템 값 세팅
```

4.6 웹서비스의 보안 대책

웹서비스는 기밀성, 무결성, 가용성, 인증과 부인봉쇄와 같은 보안요소들을 만족하여야 한다. IBM, 마이크로소프트, 베리사인, BEA시스템스, RSA시큐리티 등은 2002년 12월에 애플리케이션, 하드웨어, 최종 사용자간에 안전한 웹서비스 거래를 가능하게 해주는 'WS-보안(WS-Security)' 하부 규격을 발표하였다. 이 WS-보안 규격을 기반으로 하는 웹서비스 보안대책은 아래 표 1과 같다.

기밀성의 보장은 아래와 같이 SOAP Body안에 기밀성이 요구되는 데이터를 AES, SEED, Triple-DES, Blowfish, DES와 같은 암호 알고리즘을 이용하여 암호화하는 것이다.

표 1. 웹서비스 보안대책

기밀성	무결성	가용성	인증	부인봉쇄
SOAP 메시지 암호화	메시지 다이제스트	방화벽	Basic/ Digest	전자서명

```
<soapenv:Envelope
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/
    envelope/" xmlns:wsse="http://schemas.xmlsoap.org/ws/
    2002/04/secext" xmlns:xenc="http://www.w3.org/2001/04/
    xmlenc#" xmlns:xsd="http://www.w3.org/2001/XML
    Schema" xmlns:xsi="http://www.w3.org/2001/XML
    Schema-instance">
 <soapenv:Header>
   <wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/
    secext">

</wsse:Security>
  </soapenv:Header>

 <soapenv:Body>
  <hiddenMethod
    xmlns="urn:myservice"></hiddenMethod>
   <xenc:BodyEncryption>
    <xenc:EncryptedMethod Algorithm="AES"
      Mode="CBC" Padding="PKCS5Padding">
     <xenc:CipherData>

      <xenc:CipherValue>uHtquAj5j1qkFdBlLl7png==
      </xenc:CipherValue>
    </xenc:CipherData>
   </xenc:EncryptedMethod>

      <xenc:Params>OpMprAiD168zOc6gyEsBUQ==
      </xenc:Params>
      </xenc:BodyEncryption>
     </soapenv:Body>
    </soapenv:Envelope>
```

인증은 아래와 같이 SOAP Header안에 포함되어진 사용자 정보를 이용한다. 이 사용자 정보는 메시지 다이제스트, 암호화를 이용하여 무결성과 기밀성을 보장할 수 있다.

```
<soapenv:Envelope
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/
    envelope/"
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/
    04/secext"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance">
<soapenv:Header>
  <wsse:Security
    xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/
    04/secext">
   <wsse:UsernameToken>
<wsse:Username>smlee</wsse:Username>
   <wsse:Password Type="wsse:PasswordDigest">
   <xenc:EncryptedData>
   <xenc:EncryptedMethod Algorithm="AES"
     Mode="ECB"
     Padding="PKCS5Padding"></xenc:Encrypted
     Method>
   <xenc:CipherData>
<xenc:CipherValue>r1mpOnhyutTlBfzLNir4NUDA4h+1K
    L93+vQ9tmMr/LU=</xenc:CipherValue>
    </xenc:CipherData>
   </xenc:EncryptedData>
  </wsse:Password>
</wsse:UsernameToken>
   </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
   <testMethod xmlns="urn:myservice"/>
  </soapenv:Body>
</soapenv:Envelope>
```

마지막으로 부인봉쇄는 아래와 같이 SOAP Header안에 포함되어진 인증서를 기반으로 보장할 수 있다.

```
<soapenv:Envelope
    xmlns:ds="http://www.w3.org/2000/09/
    xmldsig#"
    xmlns:soapenv="ttp://schemas.xmlsoap.org/
    soap/envelope/"
    xmlns:wsse="http://schemas.xmlsoap.or
/ws/2002/04/secext"
    xmlns:xenc="http://www.w3.org/2001/04/
    xmlenc#" xmlns:xsd="http://www.w3.org/2001/
    XMLSchema"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    insance">
<soapenv:Header>  <wsse:Security>
    <wsse:UsernameToken>
 <wsse:Username>smlee</wsse:Username>
    <wsse:Password
    Type="wsse:PasswordDigest">
    <xenc:EncryptedData> <xenc:EncryptedMethod
    Algorithm="AES" Mode="ECB"
    Padding="PKCS5Padding"><xenc:EncryptedMet
    hod><xenc:CipherData><xenc:CipherValue>
    r1mpOnhyvtTlBfzLNir4NUDA4h+1KL93+vQ9tmMr/
    LU=</xenc:CiperValue></xenc:CipherData>
    </xenc:EncryptedData> </wsse:Password>
    </wsse:UsernameToken>

    <ds:Signature><ds:SignedInfo><ds:Canonicaliza
    tionMethod
    Algorithm="http://wwww3.org/2001/10/xml-exc-
    c14n#"></ds:CanonicalizationMethod><ds:Signature
    Method A
gorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1
    "></ds:SignatureMethod><ds:ReferenceURI=
    "#Envelope"><ds:DigestMethod
    Algorithm="http://www.w3.org/2000/09/x
ldsig#sha1"></ds:DigestMethod><ds:DigestValue>
    EsCtII4Z1ZD+Qihz0wAJyqNkAbo=</ds:igest
    Value></ds:Reference></ds:SignedInfo><ds:Sig
    natureValue>Oe0LmYeQYTm1qjW0ul2llLE3
    ivsZugVaMwF1ZpyrF3EeosUCK91dyNukTc
    X1eso4KNuh0vYXlK4&#xDHQGzUWXGvDss
    7tKUbRKLQVJf608HOyIeA5veScA2Vxcyuz8+
    ov312VPlvCjL1eZk3C5RsveL6F9U&#x9+clw
    BtFRrfAX/5Vr4A=</ds:SignatureValue><ds:
    KeyInfo><ds:X509Data><ds:X509Certifcate>
    MIIDLDCCAtagAwIBAgIQL47voq+0GDDs
    W5Ov32ob7zANBgkqhkiG9w0BAQUFADC
    BqTEWMBQGA1E&#xDChMNVmVyaVNpZ
    24sIEluYzFHMEUGA1UECxM+d3d3LnZlcml
    zaWduLmNvbS9yZXBvc2l0b3J5L1Rl&#x;c3
    RDUFMgSW5jb3JwLiBCeSBSZWYuIExpYW
    IuIExURC4xRjBEBgNVBAsTPUZvciBWZX
    JpU2lnbiBh&#xdXRob3JpeQEFBQADQQBU
    eYzwIUjFH1rm+MCq0h+X4+lbx8kRXIA8JSI
    1wpPzq87/60WZzHfuF3kTbIRr&#x3d+tDcR/
    NUQ+TDaioC95Sw+5</ds:X509Certificate>
    </ds:X509Data></ds:KeyInfo></ds:Sinature></
    wsse:Security> </soapenv:Header>
```

4.7 향후 발전 방향

방카슈랑스 시스템은 국내에서 처음으로 기업 정보시스템에 웹서비스를 적용한 사례로써 그 의미가 매우 크다고 할 수 있다. 그러나 방카슈랑스에 참여하는 일부 보험사와 금융기관들만이 웹서비스 기술을 도입함으로써 인터페이스 시스템의 중복 개발을 완전하게 회피할 수는 없다. 또한 보험사와 금융기관간의 비즈니스 메시지가 제휴 기관별로 사적 표준으로 정의되어 있어 방카슈랑스에 참여하는 모든 기관들간의 비즈니스 메시지의 공개 표준을 제정하는 것이 필요하다.

5. 사례2(웹서비스 플랫폼)

웹서비스의 핵심은 상호연동성(Interoperability)이라고 할 수 있다. 이러한 상호연동성은 표준화 기구에 의한 기술표준 정립과 표준기반의 웹서비스를 구현하는 것을 전제로한다. 웹서비스 기술 표준은 W3C, OASIS, WS-I에 의해 주도되고 있으며, IBM, MS를 비롯한 많은 벤더들이 표준에 기반한 웹서비스 플랫폼을 제공하고 있다.

5.1 웹서비스 기술 표준화 동향

• W3C (World Wide Web Consortium)

웹서비스의 필수 기술인 SOAP, WSDL 표준에 대한 이니셔티브를 쥐고 있으며, 최근 웹서비스 코리어그래피(Choreography)라는 일종의 웹서비스용 워크플로우(Workflow) 겸 오케스트레이션(Orchestration) 기술 연구에 박차를 가하고 있다.

• OASIS

W3C가 웹서비스의 기초적인 층을 쌓고 있다

면, UDDI에 대한 이니셔티브를 쥐고있는 OASIS는 ebXML과 같이 산업환경에 절실한 XML관련 기술을 주로 표준화하고 있다.

• WS-I

운영 플랫폼, 구현 언어에 독립적으로 웹서비스의 상호운영성을 보장하기 위한 기본수칙(Basic profile), 기본수칙에 따른 실례인 Sample application, 기본수칙을 검증할 수 있는 Test kit에 대한 표준화 작업을 주로 하고 있으며, .NET과 J2EE의 대표주자인 MS와 IBM의 적극적인 참여로 웹서비스 상호 운영성 문제를 해결하고 있다.

위와 같은 표준제정기구들을 통한 기술 표준 정립외에도 벤더들에 의해 개발되어 WS-I나 W3C의 영향력과 권위를 넘어서고 있는 기술표준들도 있다. 대표적인 예가 OASIS가 제정한 BPEL (Business Process Execution Language)을 웹서비스용으로 발전시킨 BPEL4WS이다. 웹서비스가 B2C보다는 B2B에 더욱 적합하다는 것이 기정사실화 되면서, 웹서비스가 BPM(Business Process Management)의 틀 안에서 운영되는 비전을 제시할 필요성이 제기되었고, MS와 IBM이 발빠르게 기존의 BPEL을 도입하여 BPEL4WS를 제정한 것이다.

현재 웹서비스는 그림 8과 같은 단순 웹서비스 (Simple Web Services)로부터 그림 9와 같은 복합 웹서비스 (Complex or Business Web Services)로 발전하고 있다. 즉 트랜잭션 처리나 메시징의 신뢰성, 보안 문제에 대한 표준이 제정되고 있고, 이에 대한 벤더들의 지원도 활발해지고 있다.

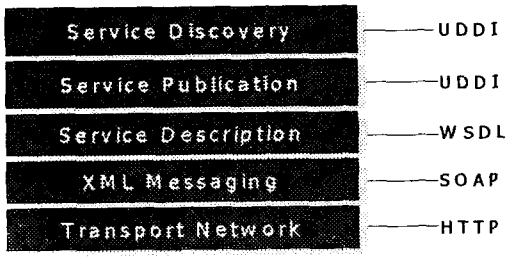


그림 8. Simple web services stack

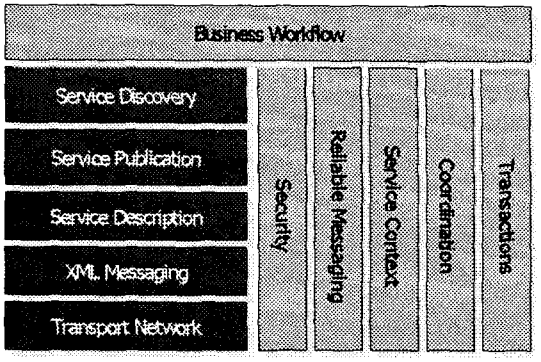


그림 9. Business web services stack

5.2 주요 웹서비스 사업자 동향

웹 서비스를 위한 주요한 소프트웨어를 개발, 공급하는 사업자는 MS의 닷넷(.Net), Sun의 선원(SunOne), IBM 의 웹스피어(WebSphere), 그리고 TmaxSoft의 웹인원(WebInOne) 등이 있다.

MS는 웹서비스를 가장 먼저 개발자와 사용자들에게 선보였으며, 사용자 친화적인 환경 제공 능력과 클라이언트 플랫폼인 윈도우를 완전히 장악하고 있다는 장점을 십분 활용, 웹서비스를 쉽고 편하게 호출하고 가공할 수 있는 환경을 제공하는데 총력을 기울이고 있다. 또한 윈도우즈 플랫폼이라는 태생적 한계를 극복하기 위하여, 상호연동성(Interoperability) 확보에 많은 노력을 기울이고 있다.

MS와 더불어 웹서비스 시장을 주도하고 있는 IBM은 기존의 WAS(Web Application Server)시장의 우위를 통해 다수의 기업 고객을 확보하고 있으며, 웹 서비스도 이러한 자사의 강점을 이용해 확대하고 있다. IBM 은 기업의 웹 어플리케이션으로 사용되고 있는 웹스피어 제품군에 웹 서비

스 기능을 추가하여 웹스피어 포탈, 웹스피어 스튜디오, 웹스피어 스튜디오 워크벤치 등을 개발하여 운영 중에 있다. 또한 최근 오픈소스 소프트웨어인 이클립스(Eclipse)를 통해 MS의 VS.NET에 대응한 WSAD(WebSphere Studio Application Developer)를 발표해 J2EE 개발자들이 손쉽게 웹 서비스 어플리케이션을 개발할 수 있도록 제공해 주고 있다.

Sun은 웹 서비스 시장의 초기 진입단계에서 MS나 IBM의 기세에 밀려 매우 고전하고 있지만, AOL, HP 등 다수의 기업이 가입하여 운영 중인 '자유연합 (Liberty Alliance)'을 통해 시장확보에 힘쓰고 있으며, 그 대안으로 자사의 웹 서비스 개발자 도구인 '디벨로퍼스튜디오'와 iPlanet의 어플리케이션 서버를 포함한 새로운 솔루션에 주력하고 있다.

TmaxSoft는 IBM과 웹 어플리케이션 부문에 치열한 선두다툼을 하고 있으나, 웹 서비스 분야는 IBM에 비해 뒤늦게 사업에 진출하였다. 그러나 자사의 웹 어플리케이션 서버인 제우스(JEUS)에 웹서비스 기능을 추가하여 웹서비스를 지원하고 있으며, 기업내 다양한 시스템과의 연계솔루션을 제공하여 기존 어플리케이션을 쉽게 웹서비스화 할 수 있다는 강점을 지니고 있다. 최근에는 개발자를 위한 '웹인원스튜디오'를 통해 웹서비스 통합 개발환경을 제공하고 있다[10].

5.3 웹서비스 플랫폼

본 절에서는 웹서비스 구현 플랫폼에 대한 사례로서 TmaxSoft의 WebInOne 플랫폼 아키텍처, 웹서비스 지원기능, 특징을 소개한다.

아래 그림 10은 기업환경에서 요구되는 복합 웹서비스를 구현하기 위한 일반적인 아키텍처를 표현한다.

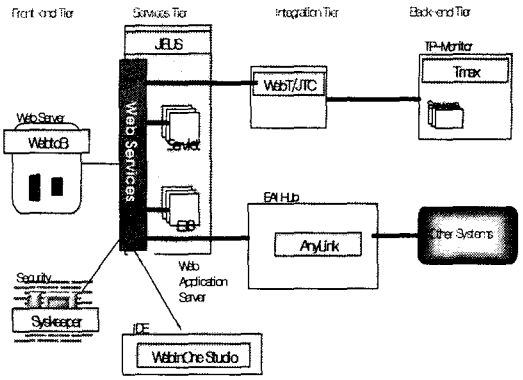


그림 10. WebInOne Business Web Services 플랫폼

웹서비스는 기업내 다양한 서비스를 통합한 형태인 포탈시스템 구성과 외부에 기업내 특정 서비스를 제공하는 경우에 많이 사용되고 있다. 업무처리 어플리케이션은 대부분 기존의 Host 시스템이나 Client/Server 시스템 상에 다양한 언어 및 형태로 구현되어 있으며, 이를 웹서비스로 제공하기 위해서는 그림 11과 같은 기존 어플리케이션을 웹서비스 형태로 랩핑(Wrapping)하는 절차가 필요하다.

기존 어플리케이션을 랩핑하는 웹서비스는 웹서비스 프로토콜을 이해하고 처리할 수 있는 서비스 계층(Services-tier)에 배포(Deploy)되어 동작하며, Integration tier의 다양한 gateway, connector를 통해 Back-end tier의 어플리케이션 또는 데이터와 연동한다. Front-end tier에서는 다양한 프로토콜을 통해 외부 클라이언트로부터의

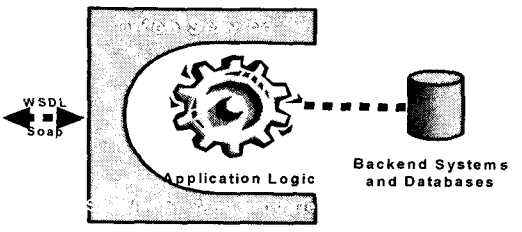


그림 11. 웹서비스 랩핑

요청을 받아들여 서비스 계층의 웹서비스를 호출하게 된다. 이러한 구조를 통해, 기업은 기존 어플리케이션에 대한 수정없이, 기업내 서비스를 통합하는 포탈시스템을 구축하고 외부에 특정 비즈니스 서비스를 제공할 수 있다.

웹서비스는 크게 .NET 기반의 구현과 J2EE 기반 구현으로 나눌 수 있으며, 기업 전산환경의 특성에 따라 적합한 기반 플랫폼을 선택하면 된다[8,11]. 웹서비스 플랫폼에서 지원하는 웹서비스 기능은 제품별로 조금씩 차이가 있으나, 표 2에서는 J2EE 기반 구현 환경을 제공하는 TmaxSoft의 WebInOne 플랫폼에서 제공하는 주요 기능을 중심으로 살펴본다.

웹서비스 엔드포인트는 웹서비스로 구현된 특정서비스를 제공하는 컴포넌트이며, WSDL로 그 위치가 기술되어 클라이언트로부터 접근되어 질 수 있다. J2EE 환경에서 현재 지원되는 웹서비스 엔드포인트는 Servlet, Stateless EJB이며, WebInOne에서는 Message Driven Bean(MDB) 형태의 Endpoint도 지원함으로써, 기업환경에서의 Asynchronous한 메시징 처리 방식을 지원하고 있다.

표 2. 웹서비스 지원 기능

표준 스펙 지원	SOAP 1.1 WSDL 1.1 UDDI 2.0
J2EE 표준 API 지원	JAX-RPC 1.0 SAAJ 1.1 JAXR 1.0
웹서비스 엔드포인트 (Endpoint)	Java 클래스 (Servlet) Stateless Session EJB Message Driven Bean (MDB)
웹서비스 저장소	UDDI Client Private UDDI Registry
보안	HTTP Authentication SSL XML Digest, XML Encryption 보안관리툴과 연계한 보안 기능
개발환경	WSDL→Java stub 생성 기능 Java→WSDL 생성 기능 모니터링 기능 Packaging, Deploy 환경 지원
복합 웹서비스	Reliable Messaging WS-Security

웹서비스를 개발하는 방법은 크게 Bottom up approach와 Top down approach로 나눌 수 있다. Bottom up approach는 기존 어플리케이션을 웹서비스화할 때 적용되는 방법으로 기존 어플리케이션 코드로부터 WSDL을 작성하여 웹서비스를 제공하고, Top down approach는 WSDL로부터 웹서비스 코드를 작성하는 방식이다. 이러한 개발방식을 지원하기 위해 WebInOne 플랫폼에서는 Java beans로부터 WSDL을 자동 생성하는 기능과 WSDL로부터 Java Stub을 자동생성하는 기능을 제공하여 개발생산성과 편이성을 높이고 있다.

기업환경에서 웹서비스 활용이 확산되면서, 단순 웹서비스로부터 복합 웹서비스로의 진화가 빠르게 진행되고 있다. 웹서비스 구현 및 운영환경을 제공하는 플랫폼에서도 복합 웹서비스를 위한 Security, Reliable Messaging, Coordination, Transaction 처리를 위한 기능이 빠르게 개발, 제공되고 있는 추세이다.

6. 결 론

웹서비스 도입에 주저하는 기업에서는 성공한 웹서비스 도입 사례의 소개가 좋은 모델이 될 수 있다. 본 논문에서 소개하는 웹서비스 응용사례는 자연스럽고 효율적으로 e-비즈니스 환경으로의 기업환경 전환을 검토하는 현장에 기술적으로 필요하게될 많은 부분이 이미 개발되어 제공되고 있고 활용되고 있는 것을 제시하고 있다. 이렇게 웹서비스가 어디에 어떻게 사용되는지에 맞춰 웹서비스를 지켜보면 이 기술의 앞으로의 전망이 매우 홍

미로울 것이다. 웹서비스의 문제점으로 거론되는 보안, 트랜잭션 속도, 트랜잭션 수 등에 관한 내용은 앞으로 연구과제로 남긴다.

참 고 문 헌

[1] 김형철, "우리은행 방카슈랑스 시스템구축," 동양시스템즈 기술연구소, 2003. 10. 9.

[2] 박대연, "웹서비스 구현을 위한 강력한 통합 구축 솔루션 : WebInOne", TmaxSoft, 2002.

[3] 오찬주 외, "서비스 지향 웹공학, 아키텍쳐, 플랫폼에 관한 워크샵," 한국소프트웨어공학협회, 2003. 10.

[4] 이한수, "웹 서비스 실전 프로그래밍," 한빛 미디어, 2002. 12.

[5] 장덕성 외, "웹기반 소프트웨어공학," 한국소프트웨어공학협회, 2003. 7.

[6] 정지훈, "웹 서비스," 한빛미디어, 2002. 6.

[7] 홍은주, "웹서비스 현황 및 향후 방향," 삼성 SDS IT REVIEW, 2002. 7.

[8] Chad Vawter & Ed Roman, "J2EE vs. Microsoft.NET A comparison of building XML-based web services," The Middleware Company, 2001. 6.

[9] TechNet Team, "Research for Web Services Technology," Tong Yang SYSTEMS Corp. 2001. 9.

[10] TmaxSoft, Internal technical materials

[11] _ , "Application Server and Web Services Benchmark," The Middleware Company, 2002. 10.

[12] _ , "Web Service Enhancement," TY Systems #26.

[13] SOAP Version 1.2 Part 0:Primer(http://www.w3.org/TR/soap12-part0/)

[14] UDDI(http://www.uddi.org)

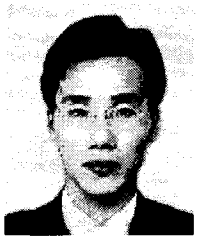
[15] Web Services(http://www.w3.org/2002/ws/)

[16] Web Services Description Language Version 1.2(http://www.w3.org/TR/wsdl12/)

[17] XML(http://www.w3.org/XML/)

김 성 익

• 1994년 한국과학기술원 기계공학과(공학사)
• 1996년 한국과학기술원 경영공학전공(공학석사)
• 1996년~2002년 동양시스템즈 기술연구소
• 2002년~현재 TmaxSoft 책임컨설턴트
• 관심분야 : S/W 아키텍처, 웹서비스, Ubiquitous computing
• E-mail : sothis@tmax.co.kr



오 찬 주

- 1993년 명지대학교 수학과(이학사)
- 1998년 광운대학교 소프트웨어공학과(이학석사)
- 1997년~현재 동양시스템즈 기술연구소 책임연구원
- 관심분야 : S/W 아키텍처, S/W 플랫폼(J2EE, .NET), 개발방법론
- E-mail : cjoh@tysystems.com

최 영 미

- 1979년 이화여자대학교 수학과(이학사)
- 1981년 이화여자대학교 대학원 전산학전공(이학석사)
- 1989년 Sydney University 전자계산학과(Visiting Scholar)
- 1993년 아주대학교 컴퓨터공학과(공학박사)
- 2001년 University of Pittsburgh 정보과학과(객원교수)
- 1994~현재 성결대학교 멀티미디어학부 부교수
- 관심분야 : 인공지능, 멀티미디어에이전트, HCI
- E-mail : choiym@sungkyul.edu