특집

웹 서비스, 서비스 지향 아키텍처와 ASP

홍 영 준* 김 명 호**

목 차

1. 서 론

CRM이나SCM과 같이 기업 고객을 대상으로 한 서버 기반의 어플리케이션의 제공과 관리사업 모델은 대형 기업 시장이 점차 포화 상태에 이름에 따라 자연스럽게 중소 기업 시장으로 눈을 돌리게 되었다. 이러한 추세는 전통적인 기업 어플리케이션 업체들뿐 아니라 대부분의 플랫폼 업체들에게로까지 확산되고 있다. 그러나 대형기업 시장을 위한 직접적인 제품 개발, 영업, 커스터마이징 방식은 지속적으로 고비용 구조를 가질 수밖에 없으며, 결과적으로 몇 개의 패키지가격을 낮추는 것만으로는 중소기업 시장에 진입하는 것이 쉬운 일이 아닌 것이 판명되었다.

매스컴의 뜨거운 관심 속에 출발한 ASP (Application Service Provider) 사업 모델은 데스크 탑 어플리케이션뿐만 아니라 기업들에게 인터넷을 근간으로 하는 기업 어플리케이션을 효과적으로 판매할 수 있는 새로운 기회를 제공한다. 본고에서는 ASP 사업 모델이 더욱 발전하기 위한 기술적 대안으로서의 웹 서비스를 소개하고, 웹 서비스 기술 동향에 대해 보다 구체적으로 다루도록 하겠다.

* 한국마이크로소프트 R&D
** 한국마이크로소프트 National Technology Officer

2. 현재 ASP 어플리케이션의 문제점

ASP는 기업 역량 집중화, 초기 투자 비용 감소, 성능과 유연성 등에 크게 기여할 수 있다. ASP는 기업이 자체적으로 어플리케이션을 개발하기보다 외부의 전문적인 개발, 관리, 운영 인력의 아웃소싱을 가능하게 함으로써 고객이 보다 자신의 역량에 집중할 수 있도록 한다. 특히 고가의 기업 어플리케이션 패키지 및 시스템 도입 비용 및 위험 요소를 제거할 수 있어서 투자에 대한 예측을 가능하게 하는 효과도 있다.

또한 전문화된 ASP 업체로부터 최상의 기능을 가진 제품을 빠른 시간 내에 도입할 수 있고, 기능 업그레이드와 관련된 서비스를 지속적으로 보장받음으로써 기업 어플리케이션의 유연성과 성능을 최상으로 유지할 수 있다는 장점도 있다. 그러나 현재 ASP 어플리케이션들이 기반으로 하고 있는 기술들은 이와 같은 ASP의 궁극적인 장점들을 반감시키고 있다.

메타 그룹에 따르면 기업들이 기업 어플리케이

션 패키지 구입 시 최소 10% 기능을 커스터마이징 하며, 이에 따른 보조 비용이 초기 도입 비용의 2-3 배에 달한다[1]. 그리고 이미 많은 기업들은 내부적으로 크고 작은 컴퓨팅 시스템을 운영하고 있으며, 이로 인해 고객은 ASP 서비스로 제공받은 기업 어플리케이션들과 내부 전산 시스템과의 통합에 대해 점점 더 많은 요구를 하고 있다. 이러한 요구는 기업의 주요 자산을 전적으로 아웃소싱에만 의존할 수 없다는 불안감과 결부되어 더욱 크게 부각되고 있다.

그런데 단순한 인터넷을 통한 원격 배포 방식의 ASP 어플리케이션들은 이러한 통합 및 커스트마이징에 따른 초기 비용과 시간, 그리고 이로 인해 복잡도가 증가한 시스템의 운영 비용에 대한 해결책을 제시하지 못하여 기대와는 달리 ASP 도입 효과를 반감시키고 있다.

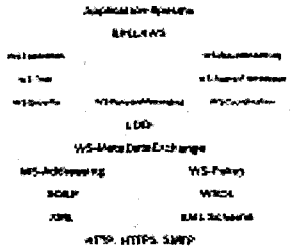
이러한 문제들은 근본적으로 소프트웨어 재사용의 단위를 확장하고, 통합성을 증진시킴으로써 해결할 수 있는데, 기존 ASP 어플리케이션들이 선택한 기반 기술들은 이런 면에서 큰 장점을 제공하고 있지 못하다. 현재 대부분의 국내 ASP 어플리케이션들이 사용하고 있는 COM, CORBA, Java와 같은 기술들은 비교적 최근의 컴포넌트 기술이라 할 수 있다. 그러나 인터넷과 같은 개방형 네트워크에서 발생할 수 있는 보안이나 신뢰성, 그리고 방화벽 등에 취약하거나 사실상 업계 표준으로 받아들여 지기는 힘든 기술이라는 단점이 있다. 결국 이러한 기술들은 대부분 어플리케이션 내부 구현 기술로만 사용되며 고객에게는 웹 페이지나 애플릿 정도만을 전송하고 있는 실정이다.

3. 웹 서비스

웹 서비스는 인터넷과 통합, 그리고 네트워크에서 접근 가능한 서비스를 목표로 설계되었으며, 플랫폼 업체로부터 기업 어플리케이션 업체에 이르기까지 광범위한 지지를 받고 있다. 웹 서비스의 이러한 특징들은 소프트웨어 재사용의 단위를 서비스 수준까지 확장하며, 통합을 촉진할뿐 아니라 서비스 제공 위치 및 제어에 대해 보다 유연한 구성을 가능하게 한다.

웹 서비스에 대한 다양한 정의가 있지만 본 고에서는 가장 광범위하게 인정되고 있는 "WSDL (Web Services Description Language)로 기술되고, HTTP로 전송되는 SOAP 메시지 교환 및 처리를 위한 분산 시스템"이라는 정의를 사용하도록 하겠다[2].

웹 서비스는 서로 다른 조직 내의 여러 하부 기술 구조에서 동작하는소프트웨어 컴포넌트를 통합하는 현실적인 대안으로 EAI 및 B2BI 등 다양한 어플리케이션 통합 시나리오에 사용되고 있다. 이와 같이 웹 서비스가 어플리케이션 통합을 위한 현실적대안으로 인식되는 이유는 기존 기술이 제공하지 못하는 특별한 기능을 제공해서라기보다는 이들이 인터넷에 기반을 두었으며, 유수의 상업 소프트웨어 업체들, 표준 기구, 연구 기관들의 합의를 도출하는 데 성공하였기 때문이다. 웹 서비스를 고려함에 있어 반드시 생각해 볼 것은 인터넷을 매개로 한 다양한 플랫폼 혹은 비즈니스간의 통합에 대한 요구 사항이다. 따라서 웹 서비스는 현재뿐 아니라 미래의 ASP 사업 모델에 매우잘 부합된다고 할 수 있다.



(그림 1) 웹 서비스 표준 스택

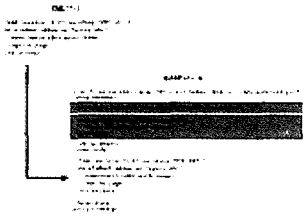
그러나 단순한 구조의 웹 서비스는 보안, 트랜잭션, 신뢰성이 요구되는복잡한 비즈니스 시나리오에 사용하기는 역부족이다. 따라서 웹 서비스 관련된 새로운 표준들이 계속해서 정의되고 있으며 아래와 같이 웹 서비스 표준 스택을 형성한다.

위 (그림 1)은 현재 논의 되고 있는 표준을 나열한 것은 아니다. 그림에서 볼 수 있듯이 많은 표준이 계속하여 등장하고 있기에 이들을 보다 체계화할 필요가 있다. (그림 1)에 표시한 웹 서비스 관련 표준을 역할별로 정리하자면, 메시징 표준인 XML/SOAP/WS-Addressing, 메타 데이터 표준인 XML Schema/WSDL/WS-Policy/UDDI, 보안 표준인 WS-Security/WS-Trust/WS-Federation, 신뢰성 메시지 표준인 WS-RelaibleMessaging, 트랜잭션 표준인 WS-Coordination/WS-Atomic Transaction/WS-BusinessActivity, 웹 서비스 컴포지션 표준인 BPEL4WS 로 나눌 수 있다. 이들 중 XML, SOAP, XML Schema, WSDL, UDDI 등은 현재 기본적인 웹서비스 표준들로 가장 널리 인정되고 있으며, 추가적인 표준들은 WS-*로 총칭하기도 한다.

표준들이 계속 개발되고 있기에 자칫 표준의 혼란에 빠질 수 있어서 이들을 바라보는 적절한 시각이 필요하다. 본 고에서는 웹 서비스와 관련된 표준에 대한 개략적인 소개와 이들이 어떻게 상호 연동될 수 있는 지에 초점을 맞추도록 하겠다.

3.1 웹 서비스 기본 표준

이 기종에서 동작하는 어플리케이션들 간의 메시지 전달을 위해 XML을 이용하는 것은 당연한선택이라 할 수 있다. 그러나 XML 자체는 정보나 데이터 자체를 전송하기에는 충분하지만, 어플리케이션 메시지 전달에는 종단점 정보, 보안, 트랜잭션과 같은 기술 하부 구조를 위한 컨텍스트 정보 역시 필요하다. SOAP 표준은 이를 위해 전송하고자 하는 정보 자체와 컨텍스트를 분리하는 위한 프레임 구조를 제공한다. 이를 위해 SOAP 표준은 컨텍스트 정보를 담기 위한 헤더와 데이터를 담기 위한 바디로 구성된 SOAP 봉투(Envelop), 그리고 HTTP, SMTP과 같은 전송 프로토콜을 이용해 SOAP 메시지 전달하기 위해 필요한 바인딩 규칙을 정의하고 있다. 현재 SOAP 1.2가W3C 권고안으로 공개되었는데, 이전 1.1 표준까지 Simple Object Access Protocol의 약어로 SOAP을 사용하여 왔으나 1.2부터는 더 이상 이 약어를 사용하지 않고 있다[3]. 이는 SOAP이 더 이상 원격 객체 시스템을 접근하기 위한 프로토콜만은 아니라는 점을 시사하고 있다.



(그림 2) XML 문서와 SOAP 메시지

XML Schema역시 문서의 구조를 기술하는 데는 적합하지만, 어플리케이션 간 메시지 교환은 이러한 문서 구조 외에도 메시지 교환 방식, 인코딩 방식, 전송 방식에 대한 명세 기술이 추가적으로 요구된다. 이를 위해 WSDL(Web Services Description Language)표준이 정의되었으며, 이를 이용해 XML Schema형태의 메시지 스키마 정보, 메시지 교환 방식(one-way, request/response, solicit-response, notification), 메시지 교환에 따른 전송 프로토콜 및 인코딩 방식, 종단점 정보를 기술할 수 있다[4].

UDDI(Universal Description and Discovery Interface)은 WSDL과 같은 서비스의 메타정보를 중앙 집중적으로 관리하기 위한 표준이다. UDDI 표준을 통해 WSDL은 서비스 제공자의 프로파일

과 함께 UNSPSC(United Nations Standard Products and Services Code System)과 같은 산업 표준 분류 별로 저장될 수 있으며 검색 역시 가능하다[5]. 이러한 정보 저장과 검색은 모두 UDDI API로 정의된 웹 서비스를 통해 접근할 수 있다.

3.2 WS-* 표준

앞 절에서 웹 서비스 기본 표준 소개에서 언급한 SOAP 표준은 메시지의 프레임 구조만을 정의하고 있으며, 기술 인프라에 관련된 컨텍스트 정보 전달을위한 구체적인 메커니즘을 정의하고 있지 않다. 이들은 WS-* 로 총칭되는 표준들에 의해 개별적으로 정의된다. 보다 구체적으로 말하자면 자신의 역할을 수행하기 위해 필요한 구체적인 SOAP 헤더의 종류와 값, 그리고 이들의 기술적 의미를 정의하는 것이 WS-* 표준들이다.

WS-* 표준들은 마이크로소프트와 IBM의 주도와 기타 업체들의 협력을 통해 작성되고 있으며, 몇몇 표준들은 OASIS와 같은 표준 기구에 제출되었다. 이들은 또한 업계(VeriSign, RSA, SAP, BEA, Oracle 등)의 강력한 지지를 받고 있어 사실상 산업 표준으로 정착되고 있다.

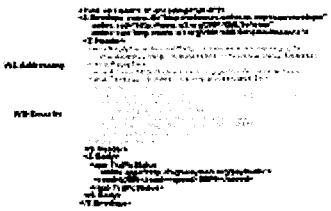
이러한 표준을 정의하기에 앞서 마이크로소프트와 IBM은 2001년, 향후 개발될 WS-* 표준들에 대한 설계 원칙과 가이드라인을 포함한 Web Services Framework를 발표하였으며, 마이크로소프트는 이를 GXA(Global XML Web Services Architecture)라 지칭하였다[6][7]. 2002년 W3C에 WSA(Web Services Architecture) WG가 구성되었는데, 이는 사실상 GXA나 Web Services Framework와 모두 동일한 것이다[2].

WSA는 모듈화, 범용 표준, 분산 관리 및 협업, 기존 표준 기반이라는 네 가지 설계 원칙을 담고 있는데, 현재 발표된 모든 WS-* 표준들은 이 설계 원칙을 따르고 있다. 이 네 가지 설계 원칙이 모두 중요하겠지만, 이중 모듈화의 원칙은 WS-*의 전체 구조를 파악하기 위해 보다 상세히 살펴볼 필요가 있다[7].

3.2.1 모듈화의 원칙

WS-* 표준들은 모듈화의 원칙을 통해 컴포넌트화 될 수 있다. 즉, 각 WS-*는 자신의 역할만을 정의하고 있으며, 다른 표준에 의존하기 보다 그 자체만으로도 사용될 수 있도록 설계되었다. 또한 메시지의 의미를 손상시키지 않고 새로운 표준이 요구하는 구성 요소를 메시지에 삽입하는 것이 가능하도록 설계되었다. 이를 통해 WS-* 표준들은 기존 소프트웨어 하부구조의 변경 없이, 개별 표준을 선택적이고 점진적으로 활용하는 것이 가능하며, 각 표준이 서로에 영향을 끼치지 않고 독립적으로 발전할 수 있다.

예를 들어 WS-Coordiation이나 WS-Atomic Transaction은 나머지 WS-* 표준들에 독립적이다. 따라서 보안 및 트랜잭션 처리를 위해서 WS-Coordination, WS-AtomicTransaction 표준, 그리고 HTTPS와 같은 보안 전송 프로토콜을 사용할 수 있다. 그러나 이후 필요에 따라기존 솔루션이나 어플리케이션의 변경 없이 WS-Security와 같은 표준을 추가할 수 있다. 즉, 필요에 따라 표준 자체를 소프트웨어 컴포넌트처럼 취사선택할 수 있도록 하며, 소프트웨어 업체들은 단일 제품 대신, 다양한 영역에서 이들을 지원하는 것이 가능하다.



(그림 3) 모듈화의 예

위의 (그림 3)은 SOAP 메시지에 WS-Addressing 표준과 WS-Security 표준이 적용된 예로서, 각 표준들은 기존 메시지를 변경하지 않고, 독립적으로 추가가 가능하다.

3.2.2 WS-Addressing

지금까지 대부분의 웹서비스들은 종단점 주소는 HTTP 전송 프로토콜의 URL형식을 사용한다. 이들은 HTTP 전송 프로토콜을 통해 전달되며, 서비스 제공자는 이를 이용해 응답 메시지를 서비스 사용자에게 전달한다. 이러한 방식에서는 서비스 참가자의 종단점 정보가 메시지의 일부가 아닐뿐 아니라, 종단점 주소외에 부가적인 종단점 정보를 다루는 것이 쉽지 않았다.

이러한 한계로 인해 전송 프로토콜의 연결이 끊어지거나 서비스 사용자와 제공자 사이에 서비스 중개자가 있는 경우 종단점 정보를 유지할 수 없다. 물론 많은 경우 서비스 제공자와 서비스 중개자는 서로 직접 연결될 수 있지만, 트랜잭션이나 보안이 요구되는 시나리오와 같이 서비스 중개자가 필요한 경우나 서비스 요청과 서비스 응답을 받을 어플리케이션이 달라야만 하는 시나리오에서도 이러한 한계는 문제를 유발한다.

WS-Addressing은 단순한 종단점 주소 외에 다양한 종단점 정보(WSDL의 포트 및 서비스 정보, 그리고 사용자 정의 정보 등)를 포함하도록 확장이 가능한 종단점 참조 모델(endpoint reference)과 이러한 정보들이 SOAP 헤더에 결합되는 규칙을 정의하고 있다[8]. 즉, WS-Addressing은 다양한 종단점 정보를 웹 서비스 메시지에 직접 첨부할 수 있는 규칙을 제공함으로써 이러한 정보들이 전송 프로토콜에 독립적으로 사용될 수 있도록 한다.

3.2.3 WS-Policy

XML Schema와WSDL 만으로는 메시지 전송 방식 및 메시지 전송에 필요한 기술 요구 사항을 완전히 명시할 수 없다. 예를 들어 서비스를 사용하기 위해 요구되는 보안 표준이 있는지, 트랜잭션을 위해서는 어떤 표준을 사용해야 하는지 기술할 수 없다.

WS-Policy 표준은 XML Schema와 WSDL에 추가적인 서비스 보증(Service Assurance) 수준을 기술할 수 있도록 한다. 서비스 제공자는 서비스를 이용하기위해 필요한 기술 수준을 서비스 사용자에게 전달할 수 있다[9]. WS-Policy는 보안, 트랜잭션, 신뢰성이 요구되는 고수준의 서비스 통합을 위해 필수 요소라 할 수 있다. WS-Policy 프레임워크에 따라 정책은 정책 식으로 표현되며, 이들은 WS-PolicyAttachment 표준을 통해XML 요소에 바인딩되거나 WSDL의 타입 정보나 UDDI 엔티티에 바인딩될 수 있다.

3.2.4. WS-MetadataExchange

지금까지 나온 표준을 종합하면 서비스의 메타정보는 WSDL, WS-Policy, XML Schema로 표현될 수 있으며 UDDI를 통해 공개하거나 검색할 수 있다. WS-MetadataExchange 표준은 서비스 제공자에 직접 연결해 필요한 메타정보를 질의할 수 있는 웹 서비스 API와 메커니즘을 정의하고 있다. 이를 이용해 개발 툴들은 간단한 웹 서비스 참조만으로 웹 서비스 호출에 필요한 프록시를 자동으로 생성할 수 있다.

3.2.5 WS-Security

웹 서비스 보안을 위해 HTTPS나HTTP Basic-Auth와 같은 전송 프로토콜의 보안 기능을 이용하는 것이 일반적인 구현 방법이다. 그러나 이들은 기본적으로 보안에 취약하거나 점 대 점 보안성만을 제공하는 약점을 가지고 있다. 예를 들어 서비스 사용자 A로부터 서비스 B를 거쳐 서비스 C까지 메시지가 전달되어야 할 경우, A-B 혹은 B-C사이에 메시지 기밀성, 인증, 무결성을 유지하는 것은 가능하지만, A의 보안 토큰을 C까지 손상없이 보내고, 중간자인 서비스 B가 A의 정보에 불

필요하게 접근하는 것을 막을 수 없다. 즉, 종단간 보안성을 제공할 수 없다.

WS-Security 표준은 이를 위해 메시지에 각종 보안 토큰을 첨부하는 기본적인 규칙과 이를 이용해 메시지를 안전하게 교환하는 기본 매커니즘을 정의하고 있다[10].

3.2.6 WS-Trust

WS-Trust는 보안 토큰 발행, 유효성 검사, 교환에 관한 메커니즘을 정의하고 있다. 보안 토큰 발행을 위해서는 커베러스(Kerberos)의 키 분배 센터(Key Distribution Center)와 같이 서비스 참가자가 공통적으로 신뢰할수 있는 보안 토큰 발행자가 필요하다[11].

WS-Trust 역시 이를 위해 보안 토큰 서비스(Security Token Service)를 정의하고 있는데 이는 현재까지 나온 보안 기술을 기반으로 토큰 발행, 유효성 검사, 교환과 관련된 기능을 제공하는 기본 메커니즘과 웹 서비스 API로 정의된다. 그리고 STS는 서비스 소비자, 서비스 제공자와는 독립적인 별도의 웹 서비스 형태로 운영될 수 있다.

3.2.7 WS-Federation

WS-Federation 표준은 마이크로소프트, IBM, VeriSign, RSA, BEA에 의해 처음 작성되었으며, 서로 다른 보안 인증 체계를 이용하는 조직이나 분산 어플리케이션들이 단일 가상 보안 도메인을 구성해 사용자나 어플리케이션 식별자(identity) 및 기타 정보를 공유할 수 있는 모델 및 세부 사항을 정의하고 있다[12]. 또한 WS-Federation 표준은 클라이언트의 종류 별로 각기 다른 프로파일을 제공하는데, 웹 브라우저를 기반으로 한 클라이언트를 위한 프로파일(Passive Requestor Profile)과 SOAP 및 스마트 클라이언트를 위한프로파일(Active Requestor Profile)을 지원한다.

이 WS-Federation은 Liberty Alliance 표준과 목적이나 기능이 유사하며 서로 경쟁 관계에 있다고 할 수 있다. 비록 Liberty Alliance 표준이 WS-Federation 표준에 앞서 발표되었지만 현재는 특정 업체 제품군을 제외하고는 Liberty Alliance 표준을 구현한 제품이 나올 가능성은 크지 않고, 오히려 WS-Federation이 더욱 폭넓은 기술 업체들의 지지를 받고 있다. 따라서 Liberty Alliance 표준도 점차 WS-Federation 표준을 지원하게 될 것으로 예상된다.

3.2.8 WS-ReliableMessaging

메시지 교환 참여자들은 자신의 메시지가 상대방에게 정확하게 전달될 것을 신뢰할 필요가 있다. 신뢰 메시지 전달 기능은 기술·하부 구조가 메시지 전달을 책임지게 함으로써 메시지 전달에 따른 예외 처리와 설계 및 구현의 부담을 감소시킨다. 따라서 신뢰 메시지 전달은 메시지 교환 시나리오에 있어 핵심적인 역할을 수행하므로 많은 업체들이 고유한 메시지 지향 미들웨어를 출시하거나 운영체제에 이러한 기능을 탑재하였다. WS-Reliable Messaging 표준은 이들이 구현 기술에 관계없이 연동하기 위해 필요한 사항들을 정의하고 있다. WS-ReliableMessaging 표준이 지원하고 있는 전달 보증(assurance)으로는 AtMostOnce, At LeastOnce, ExactlyOnce 및 InOrder가 있다[13].

3.2.9 WS-Coordination

WS-Coordination 표준은 웹 서비스 사이의 트랜잭션 처리를 위해 필요한 트랜잭션 컨텍스트 생성 및, 트랜잭션 등록, 기본적인 트랜잭션 관리 프레임워크, 그리고 메시지에 트랜잭션 컨텍스트를 포함하기 위해 필요한 메커니즘을 정의하고 있다[14].

따라서 WS-Coordiation 표준은 크게 트랜잭션 관리를 위해 필요한 웹 서비스들과 트랜잭션 컨텍스트 전송을 위한 SOAP 헤더 인코딩 방식을 정의하고 있다. 웹 서비스 상의 트랜잭션 관리를 위해 필요한 서비스들로는 활성화 서비스(Activation Service), 등록 서비스(Registration Service), 조정

자(Coordinator)가 필요하다. 활성화 서비스는 트랜잭션 요청자에 의해 호출되며, 결과적으로 트랜잭션 컨텍스트를 서비스 사용자에게 반환한다. 이후, 동일한 트랜잭션 내에 참여해야 하는 서비스를 호출할 경우, 서비스 사용자는 자신의 메시지에 항상 트랜잭션 컨텍스트를 첨부한다. 이때 트랜잭션 컨텍스트와 함께 호출된 서비스는 등록 서비스를 이용해, 자신이 트랜잭션에 참여하고 있음을 조정자에게 알리게된다.

조정자는 트랜잭션 종료 시, 트랜잭션에 참여한 각 서비스와 웹 서비스 통신을 통해 전체 트랜잭션의 결과를 결정하게 된다. 그런데, 조정자의 구체적인 역할은 별도의 조정 프로토콜(Coordination Protocol) 표준에 의해 기술된다. 현재 발표된 조정 프로토콜로는 2PC를 지원하는 WS-Atomic Transaction 표준과 트랜잭션 보정을 이용해 장기 수행 트랜잭션을 지원하는 WS-BusinessActivity 표준이 있다[15].

3.2.10 BPEL4WS

BPEL4WS(Business Process Execution Language for Web Services) 표준은 서비스 컴포지션을 지원한다[16]. BPEL4WS 표준을 이용함으로써 서비스들의 구조뿐 아니라, 서비스들의 행위를 정의할 수 있다.

BPEL4WS 표준의 행위는 기본 액티비티(Basic Activity)와 구조 액티비티(Structured Activity)에의해 정의되는데, 메시지 수신, 메시지 전송, 웹 서비스 호출은 기본액티비티에 해당하며, 순차(Sequence), 분기(Switch), 반복(While) 등은 구조 액티비티에 해당한다. 즉, 이러한 액티비티들을 이용해 서비스의 행위를 기술할 수 있다. 그런데, BPEL4WS 표준은 서비스 구조 및 행위에 대한 기술언어이기도 하지만, 서비스 구조 및 행위에 대한 실행언어이기도 하다. 따라서 변수나 메시지 상관관계(correlation)와 같은 실행에 필요한 요소들 역시 BPEL4WS 표준에 정의되어 있다.

예를 들어 상품 주문을받는 웹 서비스의 경우, 주문 이후 거래 금액에 따라 서로 다른 프로세스를 실행하거나 서로 다른 어플리케이션에 연결해야 될 경우가 있다. 이러한 웹 서비스의 행위를 표현하기 위해 BPEL4WS 표준의 분기 구조 액티비티를 이용할 수 있다. 그러나 구체적으로얼마의 거래 금액에 따라 행위가 달라져야 하는지, 혹은 메시지의 어디로부터 금액을 계산해야 하는지 하는 것은 서비스 제공자만의 문제로 외부에 알릴 필요가 없다. 이러한 구체적인 사항들은 BPEL4WS 표준의 실행 언어 요소를 이용해 표현할 수 있으며, 작성된 코드는 마치SQL언어처럼 BPEL4WS를 지원하는 어플리케이션 서버를 통해 실행되게 되고 외부에공개되지 않는다.

비즈니스 프로세스에 관한 표준은 BPM(Business Process Management) 분야의 향후 플랫폼을 점치는 주요 요소로볼 수 있어서, 웹 서비스 및 B2B를 위해 비즈니스 프로세스 정의에 필요한 표준들은 경쟁이 매우 치열한 분야다. 이들은 크게 두 종류로 나누어지는데, 먼저 실행 및 내부 비즈니스 프로세스 기술을 위한 표준으로 발표된 BPEL4WS과, BPML(Business Process Modeling Language) 표준이 있다. 그리고 비즈니스 프로세스의 웹 서비스 인터페이스 언어로는 역시 BPEL4WS와 WSCI(Web Services Conversation Language), WSCL(Web Service Choreography Interface) 표준이 있다. BPEL4WS는 역시 경쟁관계인 마이크로소프트의XLANG과 IBM의 WSFL(Web Services Flow Language)의 결합체인데, BEA와 함께 2002년 여름 발표되었고, 이후 SAP, Siebel 등과 함께 OASIS에 제출되어 현재 WS-BPEL 기술 위원회가 구성된 상태다[17].

3.2.11 WS-I.org

WS-I.org(Web Services Interpretability Organi

zation)은 마이크로소프트, IBM 주도하에 2002년 결성되었다[18]. WS-I.org의 목표는 새로운 웹 서비스 표준을 제정하는 것이 아니라 다양한 플랫폼과 운영 체제, 프로그래밍 언어로 구현된 웹 서비스들이 상호 운영될 수 있도록 지원하는 것이다. 따라서 프로파일, 테스트 도구및 가이드라인, 그리고 예제 어플리케이션이 이 단체의 산출물이다.

웹 서비스에 관한 표준은 현재도 계속 추가되고 있는데다 앞서 소개한 것과 같이 독자적으로 사용되기 보다는 몇 개의 표준이 필요에 따라 혼용되어 사용되어야 한다. 따라서, 웹 서비스 상호 운영을 위해서는 각 어플리케이션들이 동일한 표준을 사용할 필요가 있다. 프로파일은 사용 시나리오 별로 선택된 표준들의 집합이라 할 수 있다. 즉, WS-I.org는 프로파일 수준의 상호 운영성을 보장하기위한 활동을 하고 있다. 현재 WS-I BPWG (Basic Profile Working Group)이 개발한 Basic Profile 1.0a가 발표되었으며 BSPWG(Basic Security Profile Working Group)이 활동 중이다[19]. Basic Profile 1.0의 요구 표준들은 XML Schema 1.0, SOAP 1.1, WSDL 1.1, UDDI 2.0이며, 보안을 위해서는 HTTPS와 X.509 인증서의 사용을 권장하고 있다. WS-I.org는 누구나 가입할 수 있으며, 워킹 그룹 결성을 통해 산업 분야별이나 사용자 시나리오 별로 요구되는 프로파일을 개발할 수 있다.

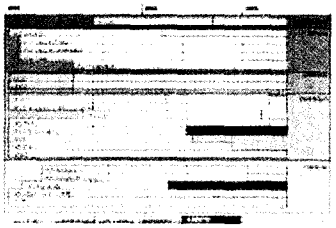
3.2.12 WS-* 표준 로드맵

많은 사람들이 미래 분산 컴퓨팅 기술의 주도권을 .NET과 J2EE의 승자가 가지게 될 것이라는 시각을 가지고 있다. 그러나 이는 잘못된 시각이며 결국은 웹 서비스가 승자가 될 것이다. 웹 서비스는 보안, 신뢰성, 트랜잭션 등 모든 분야에 걸쳐 경쟁하는 다른 디안 표준들이 있다. 이들은 모두 개방성을 표방하고 있으며, W3C나 OASIS와 같은 표준 단체와도 긴밀한 유대 관계를 맺고 있어서 승자를 점치기는 매우 힘들다. 이들은 서로 기술적인 우월성을 내세우고 있기는 하지만, 사실은 참여 업체들 사이의 사업이익에 의한 것으로 비춰지는 것도 사실이다. 그만큼 웹 서비스에대한 주도권은 각 업체들의 사활이 걸린 문제이기 때문이다.

이러한 대표적인 경쟁자로ebXML을 들 수 있다. ebXML은 범용 분산 컴퓨팅기술이라기보다는 적용 대상이 B2B 분야로 제한적이고, 적용되고 있는 기술도 대부분 메시징과 레지스트리에 국한되어 있다. UN/CEFACT 역시 향후 웹 서비스와 관련된 기술, 특히 BPEL4WS와 WS-I.org 프로파일에 보다 많은 관심을 보이게 될 것으로 보인다[20]. 결국 ebXML은 웹 서비스 표준이 성숙함에 따라 점차 많은 표준을 웹 서비스로부터 차용할 것으로 보인다.

중요한 점은 어떤 표준을 선택하든 웹 서비스는 기술적으로 더 나은 통합성을 제공한다는 사실이다. 따라서 웹 서비스 표준의 승자가 완전히 드러나기 전까지는 WS-I.org 프로파일과 같이 전반적으로 동의하는 공통의 표준 프로파일을 사용하는 것이 합리적이라 할 수 있다.

참고로 아래 (그림 4)는 CBDI Forum에서 2003년 7월에 출판된 웹 서비스 프로토콜 스택 보고서로부터 발췌한 웹 서비스 관련 표준 로드맵이다[21].



(그림 4) 웹 서비스 표준 채택

4. 서비스 지향 아키텍처

서비스 지향 아키텍처(Service-Oriented Architecture, SOA)는 어플리케이션 통합을 위한아키텍

처 측면의 관점을 제공하는 반면 웹 서비스는 서비스 지향아키텍처를 실현하기 위한 구체적 수단을 제공한다. 서비스 지향 아키텍처와 웹 서비스는 인터넷 기반의 어플리케이션 통합을 위한 매우 적절한 솔루션으로 평가받고 있다. 기존 객체 지향, 혹은 프로시저 중심 모델과 서비스 지향 아키텍처에서의 서비스와의 차이점을 위주로 살펴보기로 한다.

기존의 분산 어플리케이션 모델은 프로그래밍 모델을 그대로 반영하고 있다. 즉, 두 어플리케이션이 동일한 형 공간(type space), 프로시저 및 객체 참조 모델, 그리고 실행모델을 공유하고 있다는 가정 하에서 출발한 것들이다. 또한 원격 프로시저 호출 시 필요한 컨텍스트(상태, 보안, 종단점 등)은 대부분 묵시적으로 처리된다. 특히 객체 지향 시스템들은 형 호환만을 서비스 호환의 잣대로 사용하며, 객체 참조나 이름을 통해 원격 어플리케이션들을 연결한다. 그러나 이러한 모델은 다양한 플랫폼, 언어, 운영 체제를기반으로 하고 있는 어플리케이션들을 통합하거나, 인터넷과 같이 신뢰성이 떨어지는 통신 공간을 이용할 경우 적응력이 떨어질 수 밖에 없다.

한편 서비스는 단순한 형 대신 계약(contracts)과 스키마(schema), 그리고 정책(policy)에 의해 표현되며, 다른 어떤 전제도 하지 않는다. 여기서 스키마는 메시지의 구조와 교환 방식을 의미하는데 웹 서비스에서는 WSDL로 기술될 수 있다. 계약은 메시지 교환사이의 절차 즉, 행위(behavior)를 의미하며 웹 서비스에서는 BPEL4WS 표준 등을 통해 표현할 수 있다. 정책은 메시지들이어떤 전송 기술을 이용해 교환되는지, 혹은 서비스를 이용하기 위해 요구되는 기술 사항은 무엇인지를 나타내는 것으로 웹 서비스에서는 WS-Policy 표준을 이용해 기술될 수 있다. 그리고 이러한 메타정보들은 WS-MetadataExchange 표준 등을 이용하여 서로 교환된다. 메시지 송수신 컨텍스트 역시 메시지 자체에 포함시켜 명시적으로 공유하도록 하며, 메시지 교환은 보다안전한 비동기 방식을 기본으로 한다. 그리고 서비스는 객체 참조나이름을 통한 바인딩대신 URL이나 플랫폼 독립적인 종단정보를 기술할 수 있도록 설계된 WS-Addressing과 같은 표준을 이용해 바인딩된다.

정리하자면, 서비스는 두 어플리케이션의 기반 기술, 메시지 송수신에 필요한 정보나 컨텍스트에 대해 아무런 가정을 하지 않으며 이들을 구체적이고 명시적으로 표현하도록 요구한다.

서비스 지향 아키텍처는 분산 어플리케이션을 위와 같은 특징을 지닌 서비스로 설계하고 구현하는데 필요한 어플리케이션 아키텍처, 혹은 복잡한 서비스 통합을 효과적으로 관리, 운영, 모니터링 할 수 있는 시스템 아키텍처를 의미한다. 그러므로 효과적인 어플리케이션 통합을 위해 웹 서비스뿐 아니라 서비스 지향 아키텍처 역시 함께 고려되어야 할 것이다.

5. 결 론

ASP 사업 모델이 등장한 이후, ASP 어플리케이션은 일반 사용자를 위한 데스크 탑 어플리케이션 임대로부터 기업 어플리케이션 임대로 확장되었다. 그리고 특히 기업 어플리케이션의 경우, 초기 ASP 어플리케이션들은 기존 클라이언트-서버 기술에 기반을 두고 있었다. 그러나 이러한 기술들은 한계에 부딪혔으며, 결국 웹과 컴포넌트 기반 기술로 이전하게 되었다. 현재 ASP 어플리케이션의 일반적인 형태는 서버 어플리케이션의 중앙 집중적인 관리 및 운영, 웹 브라우저와 같은 경량 클라이언트(thin client) 사용, 그리고 인터넷을 매개로 한 배포라 할 수 있다. 이러한 방식은 초기 모델에 비해 보다 인터넷 친화적이며, 배포 및 관리가 편리하고, 궁극적으로는 성능도 뛰어나다.

그러나 현재의 ASP 어플리케이션의 형태 역시

기존 어플리케이션과의 통합과 커스터마이징, 데이터 이전에 대한 부담을 해결하기에는 역부족이며, 이는 현재 ASP 어플리케이션들이 사용하고 있는 기술의 한계이기도 하다. 기존 분산 어플리케이션모델들은 메모리 상에서 동작하는 프로그래밍 모델을 분산 환경으로 투명하게 가져오기 위해 노력해 왔다. 그러나 이들은 중앙 집중적인 통제가 불가능한 인터넷기반의 분산 어플리케이션 모델로는 부적합하다. 웹 서비스는ASP 어플리케이션의 현재와 미래를 위한 최적의 기반 기술 구조를 제공한다. 이 기술을 사용하는 ASP 어플리케이션은 보다 자유롭게 통합 및 확장, 그리고 유연한 구성이 가능하다. 웹 서비스는 인터넷 기반의 분산 환경을 보다 분명하고, 통합 가능한 방식을 다루고 있어서 ASP 사업 모델이 추구하는 방향과도 일치한다.

참고문헌

[1] Meta Group, 'ASP models face uncertainty', http://news.com.com/2009-1017-252761.html

[2] W3C Working Draft, 2003/08/08, 'Web Services Architecture', http://www.w3.org/TR/ws-arch/#whatis

[3] W3C Recommendation, 2003/06/24, "SOAP Version 1.2 Part 0: Primer", http://www.w3.org /TR/soap12-part0/

[4] W3C Note, 2001/04/1, 'Web Services Description Language (WSDL) 1.1', http://www.w3.org/TR/wsdl

[5] OASIS Standard, 2002/07/19, 'UDDI Version 2 Specification', http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv2

[6] IBM, Microsoft, 2001/03, 'Web Services Framework', http://www.w3.org/2001/03/WSWS-popa/paper51

[7] Microsoft, 2001/10, 'Global XML Web Services Architecture', http://www.gotdotnet.com/team/XMLwebservices/gxa_overview.aspx

[8] BEA, Microsoft, IBM, 2003/03/13, 'Web Services Addressing', http://msdn.microsoft.com/ws/2003/03/ws-addressing/

[9] BEA, IBM, Microsoft, SAP, 2003/05/28, 'Web Services Policy Framework (WS-Policy)' Version 1.1, http://msdn.microsoft.com/ws/2002/12/Policy/

[10] IBM, Microsoft, VeriSign, 2002/04/05, 'Web Services Security (WS-Security)' Version 1.0, http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-security.asp

[11] IBM, Microsoft, IBM, RSA, VeriSign, 2002/12/18, 'Web Services Trust Language (WS-Trust)' Version 1.0, http://msdn.microsoft.com/ws/2002/12/ws-trust/

[12] BEA, IBM, Microsoft, RSA, VeriSign, 2003/07/08, 'Web Services Federation Language (WS-Federation)' Version 1.0, http://msdn.microsoft.com/ws/2003/07/ws-federation/

[13] BEA, IBM, Microsoft, TIBCO, 2003/04/13, 'Web Services Reliable Messaging Protocol (WS-ReliableMessaging)', http://msdn.microsoft.com/ws/2003/03/ws-reliablemessaging/

[14] BEA, IBM, Microsoft, 2003/09, 'Web Services Coordination (WS-Coordination)', http://msdn.microsoft.com/ws/2003/09/wscoor/

[15] BEA, IBM, Microsoft, 2003/09, 'Web

Services Atomic Transaction (WS-AtomicTransaction)', http://msdn.microsoft.com/ws/2003/09/wsat/

[16] BEA, IBM, Microsoft, SAP, Siebel Systems, 2003/05/05, 'Business Process Execution Language for Web Services Version 1.1', http://msdn.microsoft.com/library/en-us/dnbiz2k2/html/bpel1-1.asp

[17] 'OASIS Web Services Business Process Execution Language TC', http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[18] 'Web Services Interoperability Organization', http://www.ws-i.org/

[19] WS-I, 2003/08/08, 'Basic Profile Version 1.0a', http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html

[20] UN/CEFACT, 2003/08/21, 'Press Release ECE/TRADE/03/', http://www.unece.org/cefact/press%20release%20210803.pdf

[21] Lawrence Wilkes, 2003/09, 'The Web Services Protocol Stack', http://roadmap.cbdiforum.com/reports/protocols/

저자약력

홍 영 준

1998년 동아대학교 컴퓨터공학과 (공학사)
2000년 동아대학교 컴퓨터공학과 (공학석사)
2001년 한국마이크로소프트 Evangelist
2002년 한국마이크로소프트 R&D
관심분야: 웹서비스, 엔터프라이즈 아키텍처
이 메 일: yjhong@microsoft.com

김 명 호

1984년 경북대학교 컴퓨터공학과 (공학사)
1986년 한국과학기술원 전산학과 (공학석사)
1989년 한국과학기술원 전산학과 (공학박사)
1989-1999년 동아대학교 전기전자컴퓨터공학부 교수
1997년 미국 오리건 주립대학교 교환교수
2002-2003년 모하비소프트 대표
2003년-현재 한국마이크로소프트 National Technology Officer
관심분야: 소프트웨어 공개 표준, 엔터프라이즈 아키텍처, 시스템 최적화
이 메 일: mhkim@microsoft.com