SOAP 기반의 XML 메시지 전송 시스템 개발

김용수, 주경준*, 주경수
순천향대학교 전산학과, 이비아이 솔루션*

Developing XML Message Transfer System based on SOAP

Kim Young-Soo, Kyung-Joon Ju*, Joo Kyung-Soo
Dept. of Computer Science, Soonchunhyang University
EBI Solution Inc*
E-mail : admin@kimys.pe.kr, kjju@ebisolution.com gsoojoo@sch.ac.kr

요 약

현재 많은 기업들은 수년에 걸쳐 많은 자원과 시간을 투자하여 JAVA, CORBA, COM 등을 이용해 기업시스템을 개발 해왔다. 이렇게 많은 기업들이 서로 다른 분산 컴퓨팅 기술을 가지고 개발해온 기업 시스템의 통합이 절실히 필요하게 되었다. SOAP은 JAVA, CORBA, COM 등의 분산 객체 기술에 구애받지 않는 프로토콜이며, 심지어 이들 분산 객체 기술을 전혀 사용하지 않는 애플리케이션도 SOAP을 통해 원격 프로시져 호출이나 데이터 전송을 수행할 수 있다. 본 논문에서는 이러한 SOAP의 특징을 이용한 클라이언트/서버간 메시지 전송 시스템을 구현하였다.

1. 서론

현재 인터넷은 급속도로 발전하여 모든 기업들은 인터넷을 이용하고 있으며, 이를 기업의 생산성 향상에 적용하려 노력하고 있다. 이러한 현실은 인터넷을 통해 기업과 기업, 기업과 고객의 장벽이 없어진 상황에서 서로 다른 인프라를 바탕으로 하는 시스템을 연결하는 방법이 절실한 현실이다. 그러나 JAVA, CORBA, COM 등의 주요 분산 컴퓨팅 기술은 독자적인 인프라와 기술을 사용하므로 상호 운용 성이 크게 떨어지게 마련이다.

즉, 현재의 분산 컴퓨팅을 구현하는 JAVA, CORBA, COM 등은 목적은 비슷하지만 목적을 구현하는 방법은 매우 다르므로 호환성을 기대하기 어렵다. JAVA는 RMI(Remote Method Invocation)를, CORBA는 IIOP(Internet Inter-ORB Protocol)를, COM은 DEC RPC(Remote Procedure Call)을 사용한다. 이들은 지원하는 플랫폼이 다르고 통신 수단 또한 다르기 때문에 이기종 구성요소 간의 호환이 어렵다.

각각의 기술들은 고유의 장단점을 갖고 있으며, 어느 것이 다른 것에 대해 우월하다거나 뒤진다고 말할 수 없다. 현재 기업들은 수년에 걸쳐 자신에 맞는 분산 컴퓨팅 기술에 많은 자원과 시간을 투자하여 JAVA, CORBA, COM 등을 이용해 기업시스템을 개발 해왔다. 이렇게 많은 기업들이 서로 다른 분산 컴퓨팅 기술을 가지고 개발해온 기업 시스템의 통합이 절실히 필요하게 되었다.

본 논문에서는 SOAP을 이용해 XML과 HTTP를 사용해 플랫폼에 독립적으로 메지지를 전송할 수 있는 시스템을 개발하였다. 2장에서는 관련연구와 관련 기술에 대해 알아보고, 3장에서는 SOAP 기반의 XML 메시지 시스템을 분석하고 설계한다. 4장에는 구현된 테스트 결과를 보여 주며, 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구 및 기술

2.1 관련 연구

현재까지 XML을 이용한 메시지 교환 시스템은 기존의 전통적인 EDI(Electronic Data Interchange) 시스템을 어떤방법으로 XML로 메시지를 교환할 것인가에 초점을 맞추어져 있다. EDI/XML 시스템의 성공

적인 구축은 '데이터 교환 모델'을 위하여 XML을 사용하고, '모습을 표현하기 위하여' XSL(XML Style Language)을 이용하며, 전통적인 EDI와 쉬운 통합 방안을 지니기 위하여 DTD를 사용하고, 문서중심의 조회와 처리가 가능케 하며, 타 정보 시스템과의 연동이 가능하도록 개발하는 것이다[8]. 이러한 문제를 해결하기 위하여 전자상거래 표준 기술로서 XML이 등장하게 되었으며 광범위한 영역에서 입지를 확고히 해 나가고 있다. 또한 여러 가지 장점을 가지고 있기 때문에 다양한 분야에서 적용이 되고 있다[8].

RosettaNet은 정보기술 및 전자부품의 SCM(Supply Chain Management)을 위한 XML 기반의 비즈니스 표준을 개발하기 위해 1998년에 결성된 컨소시엄으로, 350여 개 이상의 업체가 참여하고 있다. RosettaNet에서는 비즈니스 프로세스를 정의하고 데이터 교환을 위한 기술규격을 제공하고 있다.

RosettaNet에서 정의하고 있는 표준으로는 Dictionary, RNIF(RosettaNet Implementation Framework), PIP(Partner Interface Process)가 있다. RosettaNet Dictionary는 크게 비즈니스 부분과 기술 부분으로 나뉘어진다. 디렉토리는 비즈니스에서 사용되는 공통된 용어와 속성들을 표준화 한 것으로, 비즈니스를 위한 공통 플랫폼을 제공하여 개별 기업의 중복되는 투자와 노력을 절감하는 역할을 한다. RNIF는 RosettaNet 표준에 대한 시스템의 신속하고 효과적인 개발을 위한 가이드라인과 통신 프로토콜, 보안에 관련된 부분을 명시하고 있다. 즉, XML과 HTML을 사용하여 거래 파트너 사이에 정보를 교환하는 방법을 정의한다. RosettaNet에서 제공하는 표준중에서 가장 중요한 것은 PIP이며, PIP는 거래 파트너와 인터페이스 할 수 있는 비즈니스 프로세스를 정의하고 있다. PIP는 크게 6개의 클러스터로 구성이 되어 있으며, 각 클러스터는 다시 세그먼트 단위로 구분되고, 세그먼트 안에 하나의 PIP가 정의된다. RosettaNet에서는 비즈니스 모델, Dictionary, RNIF가 PIP의 입력이 되며, PIP가 거래 당사자들에게 배포되고, 각 기업에서 해당 소프트웨어를 개발하는 로드맵 역할을 한다. 각 PIP는 모든 비즈니스 로직, 메시지 흐름, 메시지 내용을 포함한다[3].

Microsoft사는 1999년 'BizTalk Initiative'라는 이름을 가지고 XML 기반의 B2B 전자상거래 솔루션을 발표하였다. BizTalk은 XML을 이용하여 기업 내부 또는 기업간 응용 프로그램 통합을 효과적으로 할 수 있는 기반을 제공해 줌으로써 보다 빠르게 전자상거래를 구축할 수 있는 방법을 제시하였다. BizTalk Initiative는 BizTalk Framework, BizTalk.org, BizTalk server의 세 가지 요소로 이루어져 있으며, 그 구성은 그림1과 같다. BizTalk을 바탕으로 하는 B2B 전자상거래 시스템에서는 거래자들이 비즈니스 문서들을 교환하기 위하여 BizTalk server를 이용한다. 기업에서 구매 요청과 같은 비즈니스 이벤트가 발생할 경우 기업의 응용 프로그램은 XML 기반 비즈니스 문서를 작성하기 위하여 이 비즈니스 이벤트에서 참조할 BizTalk Schema를 사용한다.

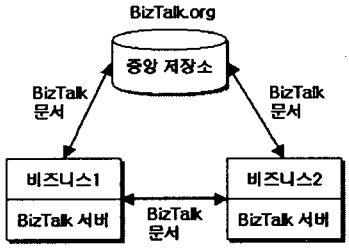

그림 1. BizTalk Initiative의 구조

BizTalk은 B2B 전자상거래와 기업 내부나 인터넷을 통하여 다른 기업간 비즈니스 프로세스를 자동화시키는 플랫폼을 제공한다. BizTalk 솔루션을 이용하여 BizTalk 지원 응용 프로그램과 BizTalk server 시스템을 구축하고, 거래자들간에 교환되어질 문서의 양식을 작성하는데 필요한 BizTalk 스키마를 결정함으로써 기업들은 전자상거래 비즈니스를 지원할 수 있게 된다[1].

2.2 관련기술

SOAP은 XML과 HTTP를 사용해 플랫폼에 독립적으로 서비스 혹은 분산 객체를 액세스하는 방법을 정의한다. HTTP는 인터넷 표준이며 누구나 어떠한 플랫폼에서도 사용할 수 있는 프로토콜이다. 그 어떤 개인이나 기업도 HTTP를 인정한다. XML은 HTTP보다 상대적으로 늦게 나온 기술이지만 최근 들어 널리 사용되며 업계 표준으로 자리잡고 있다. 또한 HTTP와 XML은 공통적으로 텍스트에 기반하고 있으므로 이들을 처리하는 소요 비용(프로세싱 시간, 메모리 등)은 상대적으로 매우 적다. 텍스트 문자열을 제어하고 TCP/IP에 접근할 수 잇는 어떠한 응용 프로그램도 HTTP를 통해 XML데이터를 전송하거나 수신할

수 있다는 것이다. 이러한 손쉬운 사용성은 이들 두 기술이 빠르게 정보기술 환경에 적응하는 발판이 되었다[2][3][4].

SOAP은 이런 XML과 HTTP를 사용함으로써 이들이 갖는 장점을 모두 포함하면서 시스템의 상호 운용성을 높일 수 있는 것이다

2.2.1 SOAP의 독립성

SOAP은 XML과 HTTP를 이용해 어떤 인터페이스의 메쏘드를 호출할 것이며, 이 메쏘드에 대한 매개변수를 알리는 역할만 담당한다. 결론적으로 말해, SOAP은 JAVA, CORBA, COM 등의 분산 객체 기술에 구애받지 않는 프로토콜이며 심지어 이들 분산 객체 기술을 전혀 사용하지 않는 애플리케이션도 SOAP을 통해 원격 프로시져 호출이나 데이터 전송을 수행할 수 있다. SOAP이 원격 객체의 구현에 대해 언급하고 있지 않으므로, 원격 객체는 어떤 프로그래밍 언어로도 구현할 수 있다[4].

2.2.2 인터넷 적용성

SOAP은 transport로서 HTTP를 사용한다. HTTP를 사용함으로써 얻을 수 있는 장점은 인터넷에서 널리 사용할 수 있다는 점이다. 실제로 SOAP은 인터넷에서 원격 객체를 액세스하기 위해 고안된 프로토콜이다. 그렇다면 기존 객체 지향 기술은 인터넷에서 문제가 있었던 것인가 살펴보도록 하자. JAVA, CORBA, COM 등이 사용하는 transport는 고유의 transport로서 RMI, IIOP, DCOM을 사용하지만, 이들 프로토콜은 TCP/IP 포트를 동적 할당하는 메커니즘을 사용하고 있다. 이것 때문에 이들 프로토콜이 방화벽을 통과하는데 문제점이 드러난다[4].

예를 들어 DCOM을 인터넷에서 사용하려면 방화벽에 일정 영역의 TCP/IP 포트를 열어 놓아야 하는데, 이것은 방화벽을 사용하지 않는 것과 다름없게 만들어 버린다. 반면 대부분의 기업 네트워크는 자사의 웹사이트를 위해 HTTP가 사용하는 포트를 열어 놓고 있다. SOAP은 HTTP를 사용하므로 아무런 문제 없이 방화벽을 통과할 수 있다. 또한 대부분의 방화벽 제품은 HTTP 헤더의 내용을 읽어 필터링을 수행할 수 있으므로 특정 인터페이스의 특정 메쏘드를 호출하는 SOAP 메시지만을 통과시키도록 설정할 수도 있다. SOAP은 HTTP를 통해 인터넷에 분산된 객체들에 접근할 수 있으며 방화벽이나 HTTPS 등의 인터넷 보안 기술을 그대로 적용받을 수 있는 것이다.

3. 시스템 분석 및 설계

3.1 SOAP 기반 XML 메시지 구조

SOAP 메시지는 두 개의 데이타 구조인 SOAP 머리글(header)과 SOAP 본문(body)을 둘러싸는 SOAP 봉투(envelope)로 구성된다. 봉투는 머리글 및 본문을 포함하는 것 외에도 이러한 데이타 구조의 정의에 사용되는 이름 공간을 정의하는 정보를 포함한다. 머리글은 생략할 수 있지만 SOAP 본문에 정의된 요청에 대한 정보를 전달하려는 경우에 사용된다. 예를 들어, 머리글은 트랜잭션, 보안, 컨텍스트 또는 사용자 프로필 정보를 포함할 수 있다. 본문에는 Web Services 요청이나 요청에 대한 응답 XML 형식으로 포함된다. 그림2는 SOAP 메시지의 상위 레벨 구조를 나타낸 것이다.

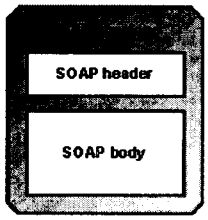

그림 2. SOAP 메시지의 구조

3.2 SOAP 기반 XML 메시지 전송 방법

클라이언트는 주문서를 서버에 보내야 하고, 서버에서는 받은 주문서를 처리하고 처리 결과를 클라이언트에 전송해야 한다. 이러한 주문서를 전송하는 데 사용되는 메쏘드가 그림3과 같다.

```
boolean PlaceOrder([in] Productcode string,
                   [in] Productname string,
                   [out] DaysToDelivery integer);
```

그림 3. 주문서 전송 메쏘드

클라이언트는 이 메쏘드를 호출할 때, 원하는 상품 코드와 이름(모두 문자열로 표현)을 전달한다. 호출이 완료되면, 메쏘드는 주문에 성공했는지를 나타내는 부울 값과 상품이 배달될 때까지 남은 예상 날짜 수를 나타내는 정수를 반환한다[5].

그림4는 주문서 작성 예이다. 위에 있는 메쏘드를 호출하면, 이 요청은 어떤 프로토콜을 사용하여 어떤 형식으로 서버에 전달될 것이다. 만약 SOAP이 사용된다면, 그 프로토콜은 HTTP이고 요청의 형식은 다음과 같습니다.

```
POST /dblabServer HTTP/1.1
Host: dblab.sch.ac.kr
Content-Type: text/xml-SOAP
Content-Length: 160
MessageType: Call

<SerializedStream>
   <PlaceOrder>
    <Productcode>01-3234</Productcode>
    <Productname>SKYMAN</Productname>
   </PlaceOrder>
</SerializedStream>
```

그림 4. 주문서

앞의 예를 보면 알 수 있듯이, SOAP은 표준 HTTP POST verb를 사용하여 전송될 수 있다. HTTP 헤더가 나타내는 대로, 이 요청은 SOAP 클라이언트에 있는 응용 프로그램 SOAP 서버로 전송된다. 메시지의 컨텐트 타입은 SOAP을 위해 새로 정의된 컨텐트 타입인 "text/xml-SOAP"이고, 메시지의 길이는 160 바이트이다. 그 다음 헤더인 MessageType은 이 메시지가 호출, 즉 SOAP 요청을 포함하고 있음을 나타낸다. 그 다음에 오는 것은 XML-정의 형식으로 표현된 요청 바디고, 바디는 XML 요소 <SerializedStream> 의 인스턴스이며, 이 요소 안에는 호출되는 메쏘드의 이름과 동일한 이름을 갖는 또 다른 요소가 있다. 그림3에서 메쏘드의 이름 PlaceOrder와 동일한 <PlaceOrder> 요소가 이에 해당된다. 또한, 이 요소 안에는 요청과 함께 전달되는 매개 변수를 나타내는 요소들이 있다. 이 요소들은 메쏘드의 [in] 매개 변수이다. 그림3에서 이들 요소 각각의 이름은 매개 변수의 이름과 일치하며, 각 요소에는 적절한 값이 들어 있다. 즉, <Productcode> 요소에는 "01-3234"이, <Productname> 요소에는 "SKYMAN"가 들어 있다. 이 간단한 XML-정의 정보에는 서버로 호출 요청을 전송하는 데 필요한 모든 것이 포함되어 있다.

SOAP 서버는 이 메시지를 수신하고 처리한 후, 그림5와 같은 응답을 반환할 수 있다.

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml-SOAP
Content-Length: 177
MessageType: CallResponse

<SerializedStream>
   <PlaceOrderResponse>
      <__return>1</__return>
      <DaysToDelivery>7</DaysToDelivery>
   </PlaceOrderResponse>
</SerializedStream>
```

그림 5. 반환형식

그림5는 성공적인 POST 요청에 대한 표준 HTTP 응답이며, 200이라는 응답 코드를 포함하고 있다. Connection 헤더는 기존의 TCP 커넥션이 닫혀야 함을 나타내며, 전과 마찬가지로 Content-Type은 "text/xml-SOAP" 이다. Content-Length 헤더 뒤에 있는 것은, 요청에서와 마찬가지로, MessageType 헤더이다. 응답에서 이 헤더의 값은 CallResponse이고, 바디가 호출의 결과를 담고 있음을 나타낸다.

요청에서와 마찬가지로, <SerializedStream> 요소에는 응답 정보가 들어 있고, 요소 바로 안에 들어 있는 요소는 호출된 메쏘드와 동일한 이름을 가져야 하고, 그 앞에 "Response"라는 단어가 붙는다. 그림5의 응답은 PlaceOrder 메쏘드에 대한 응답이므로, 이 요소는 <PlaceOrderResponse> 이다. 이 요소 안에는 메쏘드 호출로부터 반환되는 정보가 들어 있는 두 개의 요소가 있다. 첫 번째 요소인 <__return>에는 메쏘드의 반환 값이 들어 있는데, 그림5의 예에서는 주문이 성공했으므로, <__return> 요소에 1로 표현되는 부울 값 TRUE가 들어 있다. 두 번째 요소인 <DaysToDelivery> 에는 정수 7이라는 메쏘드의 out 매개 변수 값이 들어 있다. 그림5의 예에 나오지는 않았지만, SOAP는 다양한 오류 결과를 정의하여, 클라이언트가 호출이 실패한 경우 어떤 일이 발생하는지를 명확하게 판단할 수 있게 한다.

SOAP는 요청, 응답, 그리고 그 안에 포함된 정보를 표현하는 데 대해 표준 XML 기반의 규칙을 정의하기 때문에 그 성능이 강력하다. 또한, SOAP의 메시지가 표준 HTTP 요청으로 전달될 수 있으므로, 다른 분산 객체 프로토콜을 막는 방화벽과의 문제를 피할 수도 있다. 방화벽은 다른 프로토콜의 경우와 마찬가지로, 필요에 따라 SOAP 요청을 블록킹하도록 구성될 수 있다. 그러나, 다른 분산 객체 프로토콜의 경우와는 달리, SOAP 요청을 허용한다고 해서 반드시 포트를 개방해야 하는 것은 아니다.

이처럼 SOAP은 많은 기업시스템들 간의 통신 및 메시지 전송을 플랫폼에 독립적으로, 또한 방화벽이라는 장벽을 해결 할 수 있는 좋은 방안이다.

3.3 설계

3절에서 설명한 SOAP을 이용해 메시지 전송 시스템을 구축하고 SOAP 서버를 이용한 이기종 간의 통신이 가능하도록 하였다. 이를 위한 설계를 살펴보도록 한다.

3.3.1 순차 다이어그램

그림6은 SOAP을 이용한 상품 주문 시스템의 전체

흐름을 UML(Unified Modeling Language)의 순차다이어그램을 통해 표현하였다.

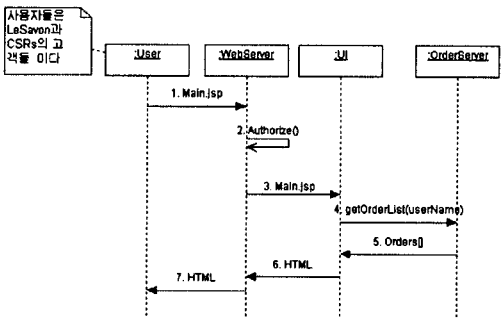

그림 6. 순차다이어그램

그림6을 보면 사용자는 첫 페이지를 요구하고, 웹서버는 사용자에게 권한을 부여한다. 권한부여 결과가 다시 UI(User Interface)로 전달되고, 이제 getOrderList(userName)를 이용해 서버를 호출하고 결과를 Order[] 의 배열 형태로 반환한다. 그리고 이것은 각각 html 형태로 반환하여 표현된다.

3.3.2 SOAP 서버 구조

SOAP 서버의 구조를 그림7의 활동 다이어그램과 그림8의 클래스 다이어그램을 통해 알아본다[5]. 그림7은 SOAP 서버의 getOrderList()를 통한 호출 과정을 UML의 활동 다이어그램을 통해 표현하였다. 그림7을 보면 클라이언트 HTTP 에서 SOAP을 이용하여 getOrderList()를 호출한다. 클라이언트에 의한 타겟 URL은 Serveur 서블릿 클래스가 된다. 요청이 받아들여 지면 getCacheData()에 의해 SOAPCache 클래스를 호출하게 되고, SOAPCache 에서는 다시 getOrderList()에 의해 RPCRouterServlet 이 호출되며 여기에서는 SOAP을 이용한 호출이 이루어진다[6].

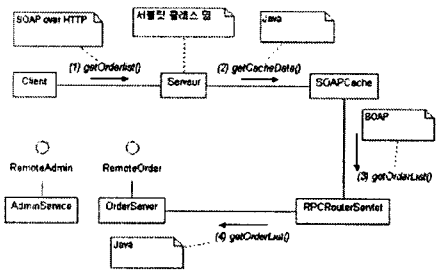

그림 7. 활동다이어그램

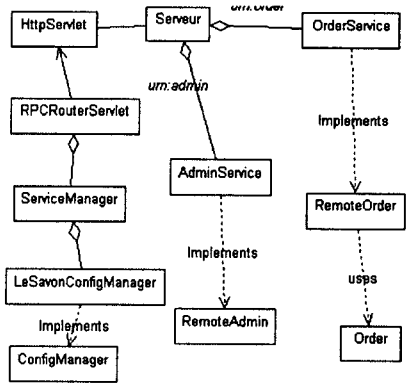

그림 8. 클래스 다이어그램

4. 구현

본 논문에서 구현된 시스템 사양으로는 Windows 2000 server, tomcat3.2.2, SOAP2.2, Xerces1.4.3 으로 구성된다. tomcat은 웹서버 기능을 담당하고, SOAP은 SOAP 서버 기능을 그리고 Xerces는 XML 파서의 기능을 수용하기 위해 사용하였다. 그림9는 메시지를 전송하고 전송한 메시지의 리스트를 확인한 화면이고, 그림10은 전송된 메시지의 처리 결과를 보이고 있다.

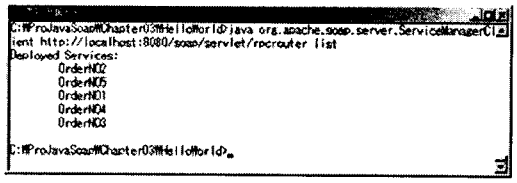

그림 9. 서버쪽 메시지 전송결과

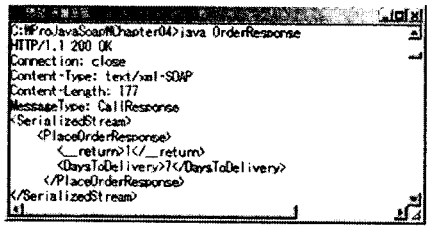

그림 10. 클라이언트 응답메시지

본 논문에서 구현한 메시지 전송 시스템을 이용한 쇼핑몰의 구조는 그림10과 같다. Presentation Tier에는 웹서버가 존재하며 웹 프로그램으로 JSP를 이용하여 Presentation을 담당하도록 한다. Business Tier에 SOAP 서버를 위치 시켜 이기종간 메시지도 처리

할 수 있도록 하였으며 데이터 베이스 접속은 JDBC를 이용한다.

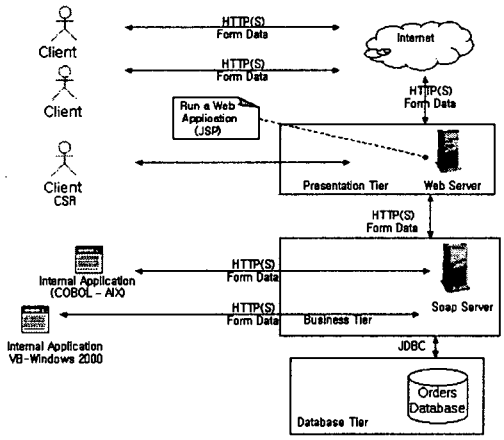

그림 11. 사이트 구조

5. 결론

현재 인터넷은 급속도로 발전하여 모든 기업들은 인터넷을 이용하고 있으며 이를 기업의 생산성 향상에 적용하려 노력하고 있다. 이러한 현실은 인터넷을 통해 기업과 기업, 기업과 고객의 장벽이 없어진 상황에서 서로 다른 인프라를 바탕으로 하는 시스템을 연결하는 방법이 절실한 현실이다

본 논문에서는 SOAP를 사용하여 서로 다른 플랫폼과 서로 다른 분산 객체 기술로 구현된 어플리케이션간의 통신을 위해 SOAP 서버를 구현하였다. 이렇게 함으로써 플랫폼과 운영체제, 애플리케이션 그리고 프로그래밍 언어를 뛰어넘는 분산 객체 프로토콜을 이용한 환경을 제공하게 된다.

[참고문헌]

[1] MicroSoft, "Biztalk Framework 1.0 Independent Document Specification" http://www.biztalk.org

[2] Extensible Markup Language(XML)1.0, http://www.w3.org/TR/1998/REC-xml-19980210

[3] Liam Quin, Open Source XML Database Toolkit, WILEY, 2000

[4] Simple Object Access Protocol(SOAP)1.1, http://www.w3.org/TR/SOAP

[5] Simple Objedt Access Protocol(SOAP), http://www.microsoft.com/korea/msdn/workshop/xml/general/SOAP_White_Paper.asp

[6] DAVID CARLSON, MODELING XML APPLICATIONS WITH UML, Wesley, 2001

[7] Bequet, Henry. Professional Java SOAP, Wrox 2001

[8] 신동규, "XML/EDI 시스템의 설계 및 구현", 한국정보처리학회 논문지 8-D권 제 2호, 2001