5-1-2001

Information sharing architecture using internet's XML and SOAP

Puskar Adhikari

Information Sharing Architecture Using Internet's XML and SOAP

Puskar Ns. Adhikari

B.S. Henderson State University
(Arkadelphia, Arkansas 1999)

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Department of Industrial and Manufacturing Engineering in the Kate Gleason College of Engineering of the Rochester Institute of Technology

May 2001

KATE GLEASON COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

MASTER OF SCIENCE DEGREE THESIS

The M. S. Degree Thesis of Puskar Ns. Adhikari
has been examined and approved by the thesis committee
as satisfactory for the thesis requirement for the
Master of Science degree.

Dr. Sudhakar Paidy, Ph.D. *Advisor*

Dr. Wayne Walter, Ph.D.

Permission granted

Information Sharing Architecture using Internet's SOAP and XML

I, Puskar Ns. Adhikari, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 05-26-2001 Signature of Author: _____

Dedication

To my Family who have taught me diligence will lead to success one day and not to ever give up

Acknowledgment

Throughout the time I have been working in this research study, many people have assisted me. It is impossible to acknowledge them all, nevertheless, I would like to specifically thank the following individuals:

Dr. Sudhakar Paidy, for all the time he dedicated to my research work and for the knowledge he has shared with me in the last two years. This thesis would not have been possible without your help.

Dr. Wayne Walter, for his support and comments on my work. I hope you enjoyed being a member of my thesis committee.

The faculty and staff of the Industrial and Manufacturing Engineering Department, not only for their academic support and feedback on my work, but for all the advise given throughout my career at RIT.

To Mr. Mark Larson and Mr. Scott Davis for the internship opportunity with Kineticworks Co. where I was able to gain my first professional experience as a Web Programmer.

To Mr. Hickey Brian from Maxim Group (Pittsburgh Office) for giving me insight on the Internet, XML and B2B in general.

To Dipesh, Prakash, Philips, Pavani, Bhairav and Elizabeth for helping me from the beginning of the thesis, proof reading, giving me feedback and support. But mostly, for your friendship, all of you have a special place in my heart.

Outline

Section **Page**

Tables and Figures

Tables and Figures

Appendix

Source Code for Shopping AsiStore Shopping cart Application
Source Code for pullxmlfromstring.dll component
Source Code for scanadotoxml.dll component

1. Abstract

Businesses should be able to share information among each other irrespective of the platform, operating systems and programming languages. Using Internet as the Information Systems Architecture has many values. Internet is affordable, easily available and is not tied to any specific vendor. Internet is simple and runs under any kind of operating system.

Information sharing across the Internet is challenging but rewarding. Data Transfer using the Internet requires structure and discipline. To integrate diverse group of systems we need specialized protocols that can connect different platforms that use different languages together. EXtensible Markup Language enables the creation of application dependent vocabulary which can be used to store data and information in a structured fashion. Simple Object Access Protocol can be used to carry information electronically from one end to the other. Simple Object Access Protocol uses the World Wide Web's eXtensible Markup Language in encoding the message contents and its Hypertext Transport Protocol in carrying the message packet.

This thesis work is focused upon sharing of information among enterprises using eXtensible Markup Language, Simple Object Access Protocol and decentralized database systems. An online Shopping cart application has been implemented using the notion of XML and SOAP. SOAP is used as a protocol to share information between AsiStore and its business partners Store1 and Store2. XML have been used as a part of the application to drive the shopping cart, which users can view on the web browser.

2. Introduction:

Information sharing has evolved from the exchange of paper and hand written documents into sharing of documents electronically within and across enterprise boundaries. For a decade or so, the consensus notion of computing has been undergoing a transformation from "computing happens in a single CPU" to "the network is the computer." (Burbeck, 2000). Remote Procedure Call (RPC) is one of the mechanisms which facilities the sharing of information by enabling a program to communicate processes running on different computers in the network. RPC requires a communications infrastructure to set up the path between the processes and provide a framework for naming and addressing (Munro, 1999). RPC is a not a new methodology and has been used over the past decade to invoke processes running at remote endpoint or in the network. There are various mechanisms for achieving Remote Procedure Call mechanism that developed along with the evolution of computers and the networks.

The earlier RPC mechanism enabled an application to invoke functions or subroutines residing on different computers in the network. In the nineties, there has been increasing use of object-oriented programming languages such as C++, Java, Visual Basic etc. This makes it necessary to invoke not only function but also objects that may be residing on different endpoints in the network. Among the most widely used object oriented RPC calls are Microsoft's DCOM and Object Management Group's Internet Inter-Orb Protocol (IIOP). We have seen architectures based on distributed object systems communicating through remote method calls and marshaled or demarshaled objects (for example, CORBA). (Burbeck, 2000).

The most widely used RPC Calls mechanism such as DCOM or Object management Group's Internet Inter-Orb Protocol (IIOP) comes to a dead end when it comes to making services communication via the Internet. 'Internet cannot guarantee what kind of client and server will be operating at either end of the connection – it can only guarantee that they are both communicating in HTTP (Borders & Dumser, 2000). The richness and the tightly coupled nature of the protocols such as DCOM and IIOP complicate the implementations and the applications that use them. DCOM and CORBA/IIOP both rely on fairly high-tech runtime environments (Box, 2000).

The use of Internet has grown exponentially over the past decade and has become an integral part in everyone's life. The Internet has revolutionized the computer and communications like nothing before. The Internet was the result of some visionary thinking by people in the early 1960s who saw great potential value in allowing computers to share information on research and development in scientific and military fields (Howe, 2001). When the Internet was first invented it was used only in research institutions and the military. The World Wide Web (WWW), which uses the Internet Protocol Hypertext Transport Language (HTTP) as its primary protocol, has been the most widely used Information System in the world today. The WWW has also tied itself with many other common Internet protocols such as FTP, SMTP, and NNTP etc.

Tim Berners-Lee invented the WWW in 1989 for the purpose of global information sharing. He wrote the first web client (browser-editor) and server in 1990. The WWW was designed with the purpose of sharing information within and across research laboratories. The intent of WWW was to display the contents hosted at a web sever, on a client browser through the use of HyperText Transport Protocol (HTTP). The contents are displayed using a set of Markup tags (HTML), which are enclosed in the web file. Today the WWW has evolved into separate Information systems, which need to work interactively with various database systems, heterogeneous components, operating systems and platforms. The growth of the browser has been beyond the imagination of the creator of this technology.

"While the 90's was all about implementing ERP systems, in the first decade of this century, the race will be to connect ERP systems," claims Peter Burke, Director of Business Development ECOutlook.com (ECOOutlook.com). The 90's have resulted in the use of a wide variety of Operating Systems and platforms than ever before. None of the existing Distributed Computing Technology as DCOM, CORBA, and Java RMI seem to have solved the issue of sharing information across platforms using different languages. It is slowly becoming realized that the Internet is the most promising Information sharing Architecture to integrate various platforms and language used within and across enterprise boundaries. The Internet is affordable, universally accepted and can be implemented and used by enterprises of any background. The Internet is a platform itself. A platform is made up of tools and runtimes. Internet tools run on all kinds of operating systems, as do the runtimes. The beauty of the net is its simplicity, its ubiquity and its lack of controlling vendor (Winer, 1998).

Simple Object Access Protocol (SOAP) was born into distributed computing technology about 18 months ago as the brainchild of a few developers and researchers at DevelopMentor and Microsoft. They created SOAP to meet the needs of developers who found distributed computing difficult to deploy with DCOM and CORBA, because firewalls typically don't pass these protocols and administrators are unwilling to open the firewalls to allow them (Marcato, 2000). The Simple Object Access Protocol (SOAP) facilitates interoperability among a wide range of programs and platforms, making existing applications accessible to a broader range of users. SOAP combines the proven Web tech-no-logy of HTTP with the flexibility and extensibility of XML (Box, 2000). The architecture of SOAP is more structured than telnet command protocols or CGI's but less complex or heavy than compared to CORBA or DCOM. SOAP can be considered to be a light weight Application Protocol, even a Remote Procedure Call mechanism built upon minimal technological requirements such as HTTP to carry message and eXtensible Markup Language to serialize its message content. The secret of the success of many other widely adopted protocols is just that: offering essential services with the lowest learning curve (Marcato, 2000). One of the reasons SOAP has chances of gaining acceptance is that it is built upon affordable and easily available Internet technologies such as HTTP and XML. The other flexibility SOAP provides is that it can be programmed using both scripting or higher level programming languages like Java and C++.

The XML used by SOAP to serialize message content is a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. Depending on whom you talk to, XML is either SGML-lite or HTML on steroids (Udell, 1999). XML is a derivative of another but more complicated form of Markup Language SGML (Standard Generalized Markup Language). XML, unlike HTML, is not a language to present data on the web browser and does not limit itself to a set of predefined tags. With XML users can define their own tags and give meaning to it. SOAP is built upon the openness of XML where the set of tags to design the message packet to be transported confides to a set of specifications (SOAP Specification). SOAP uses the WWW's HTTP Protocol to transport message from one endpoint to the other. CORBA, DCOM, and even RMI are built around live, resource-hungry client/server connections and (usually) good connectivity and bandwidth (Megginson, 2000). Since, SOAP is built upon the notion of the stateless nature of the HTTP protocol, lightweight and stateless systems can be designed. The Hypertext Transfer Protocol (HTTP) is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web (http://whatis.com). SOAP enables the exchange of information in a decentralized, distributed environment through the use of WWW's XML and HTTP in the Internet.

Today's e-businesses have brought together traditional IT and the Web in a business-to-client relationship for existing marketplaces (Smith, 2000). The traditional business-to-consumer interaction model does not always compensate for people's desire and willingness to buy products directly by visiting the stores. However, business model designed by the integration businesses electronically is different. There are economies of cost savings and economies of scale associated in doing B2B electronically and in a standard format (XML). The business-to-business arena has had enormous commercial activity over the last year and a half—linking businesses together using the common language of XML (Hertzfeld, 2001).

The next generation of e-business will leverage Internet, IT, and object technologies to create new dynamic marketplaces based on innovative business models (Smith, 2000). The next generation of e-business model assumes that businesses will develop and deploy adaptive and self-configuring applications composed of web services. Web Services describes distributed software architecture of service components that can be integrated at run time to produce dynamic and flexible applications (Fisco, 2000). They're programming-language agnostic, publish their specific function, can be discovered at design or execution time to participate in a given job, and typically interact with other Web services through XML messaging (Smith, 2000).

Web Services are a combination of Industry specific business directory UDDI (universal Description, discovery and integration), XML based communication standard SOAP, and a language to define itself, WSDL (Web Services Definition Languages). In terms of Web Services, SOAP is two things. One, it is a messaging protocol that uses XML exclusively to transport documents and procedure based content. Secondly, it is an implementation that is like BizTalk's implementation where the routing, delivery, and transactions are all guaranteed. This is achieved by embedding key elements inside an

XML message so that the evolution/devolution software (BizTalk, ebXML) can handle them correctly. The goal of business initiatives such as BizTalk.org and EbXML.org is to provide resources for learning about and using Extensible Markup Language (XML) for Enterprise Application Integration (EAI) and business-to-business (B2B) document exchange, both within the enterprise and over the Internet. However, simply implementing BizTalk or ebXML does not guarantee that businesses will ever find or ever discover each other. For businesses to discover themselves dynamically the services provided and definitions have to be defined using WSDL and registered on the global business directory UDDI. UDDI is the building block which will enable businesses to quickly, easily and dynamically find and execute transactions with one another via their preferred applications (http://www.uddi.org). UDDI is platform independent business initiative that enables enterprises to discover, describe and integrate business services using the Internet. UDDI is a global business originally initiated by business and technology leaders as Arbia, NTT, Nortel, IBM, and Microsoft. Any single industry or a particular vendor does not own UDDI. Similarly, Web Services Description Languages (WSDL) is an XML-based grammar for describing the location, functions, properties and the invocation of Web Services. The Web Services Description Language is the XML equivalent of a resume -- it describes what a Web service can do, where it resides, and how to invoke it (Glass, 2001).

The objective of this thesis work is to do a research on an Information Systems Sharing architecture and its application in the Business-to-Business World. The thesis work is focused on using the latest trend in Internet technology such as XML and its dependent protocol SOAP in sharing information across enterprise boundaries. A study has been made to explorer how XML can be used to design message packets and to transport to the other end of the wire taking advantage of the stateless nature of the HTTP protocol.

An online business store, AsiStore, has been designed as an application to illustrate the concepts of XML and SOAP. AsiStore is a virtual Computer store that sells hardware and software products. It does business with two other stores, Store1 and Store2. Store1 manufactures and supplies computer hardware products. Store2 is a distributor of Computer Software. These stores maintain their inventory on database tables. The AsiStore website is a virtual store where customers interested in computer products come to. The catalogue of products is maintained dynamically by retrieving items available from store1 and store2. The customers will not be aware that the items are actually being retrieved from the business stores of AsiStore Store1 and Store2. The costumers will be involved in a simple Business-to-Consumer application. However, AsiStore is actually using a Business-to-Business model where it will be interacting with its business partners using the SOAP messaging to retrieve information.

References

Borders, William. 'SOAP: Simple Object Access Protocol'. <**http://idm.internet.com/articles**>.

Box, Don. "A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages". <**http://msdn.microsoft.com/msdnmag/issues/0300/soap/soap.asp**>.

Burbeck, Steve. 'The Tao of e-business services'. <**http://www-106.ibm.com/developerworks/webservices/library/ws-tao/index.html?dwzone=webservices**>

Fisco, Dave. "IBM's Web Services architecture debuts". <http://www106.ibm.com/developerworks/web/library/w-int.html> (September 2000).

Glass, Graham. "The Web services (r)evolution" <**http://www-106.ibm.com/developerworks/webservices/library/ws-peer1.html?dwzone=webservices**>

Hertzfeld, Andy. 'Easier System Management'. <**http://www.futureofsoftware.net/ah0010/ah0010.asp**>

Howe, Walt. 'A Brief History of the Internet'. < http://www0.delphi.com/navnet/history.html>.

Marcato, David. "Distributed Computing with SOAP". <**http://www.devx.com/upload/free/features/vcdj/2000/04apr00/dm0400/dm0400-1.asp**>

Munro, Alistair. 'Moving to Distributed Processing Standards...Remote Procedure Call'. <http://www.ja.net/documents/NetworkNews/Issue44/RPC.html>.

Smith, Rod A. E-business Software. < **http://www.futureofsoftware.net/rs2_0010/rs0010.asp**>.

Winer, Dave. "Dave's History Of SOAP". < **http://www.xmlrpc.com/discuss/msgReader$555**> (September 25, 1999).

Winer, Dave. "SOAP meets RSS". <http://www.thetwowayweb.com/soapMeetsRss>

Winer, Dave. "RPC over HTTP via XML" <**http://davenet.userland.com/1998/02/27/rpcOverHttpViaXml**> (Feb 27, 1998).

3. Business to Business Information Exchange

3.1. B2B-Business to Business Information Exchange

The electronic business information exchange model is heavily relying upon XML, new Internet protocols as SOAP and web services. XML has gained popularity in the electronic information exchange world with its simple, extensible and inexpensive nature. XML is replacing (EDI) Electronic Document Interchange-a previous standard adopted by businesses for exchanging electronic information and document with among enterprises. EDI works well only with the top fortune five hundred companies but not among smaller businesses due to the high cost associated with it. This is the primary reason for enterprises to lean towards free and easily available technology as XML. An approach towards standardizing EDI into XML was envisioned to enable enterprises of different sizes to interact with each other in the global market. The XML/EDI did not only enable traditional EDI data to be incorporated into structured XML format but also enabled the interchange of business rules and processes among enterprises.

B2B has become the heart of electronic commerce. Electronic Marketing strategies as B2C (Business to Consumer) could never gain its full momentum in the electronic commerce world because humans can always go out and buy goods directly from the store. Electronic commerce is more than just buying and selling goods between consumers and producers on the Internet (B2C) within the same organizational boundary. The power of electronic commerce is visible only when enterprises are able to share information electronically among each other extensively. Enterprises should be able to share business information and services dynamically without major human intervention in the process.

"While the 90's was all about implementing ERP systems, in the first decade of this century, the race will be to connect ERP systems," claims Peter Burke, Director of Business Development ECOutlook.com (http://ecommerce.internet.com/solutions/). The next generation of e-business is header towards sharing of information electronically making use of newborn Internet friendly technologies as XML, SOAP and many others. The goal has been to make use of existing languages and integrate information among various platforms instead of reinventing everything from scratch. The nineties have led to the birth of so many platforms and languages. The major concern of this century has been to incorporate common standards so that the existing platforms and languages can be integrated easily to fit into businesses information-sharing architectural model.

3.2. Common Vocabulary and Standardization

XML has revolutionized the Electronic Information Sharing model as users have the flexibility of defining their own set of tags and inherit from preexisting documents. XML encapsulates data into its sets of user-defined tags in ASCII format, which can be easily interpreted any object-oriented languages and operating systems.

XML gives us the flexibility of defining our own sets of tags in the business information exchange world. The extensibility and flexibility of XML but is not the straight solution to the information-sharing model in the business world. Enterprises can use XML to represent their data using their own sets of vocabulary. However, for business partners to understand there has to be a standard information-sharing model which businesses can discover and integrate each other's vocabulary. Two major initiatives have been purposed to establish an information-sharing model among enterprises:

- EbXML.org (http://www.ebXML.org)
- BizTalk.org (http://www.BizTalk.org)

Both of the above initiatives are headed towards standardizing electronic Information sharing between businesses but with different approaches.

3.2.1. EbXML.org

EbXML.org is a joint initiative taken by United Nations body for Trade Facilitation and Electronic Business (UN/CEFACT) and the Organization for the Advancement of Structured Information Standards (OASIS) to standardize interchange of documents electronically between enterprises. The mission of ebXML.org is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties (http://www.ebXML.org). EbXML.org has been an ongoing eighteen months effort to enable a global electronic marketplace where enterprises of different sizes and locations can conduct business with each other using XML messages. The latest specification of ebXML.org has finally agreed upon using SOAP (Simple Object Access Protocol) in exchanging message between enterprises.

It is important to realize, however, that the role of ebXML is not to replicate the reliance on electronic versions of common paper documents such as purchase orders, invoices, tender requests and to offer up and develop such implementation examples (http://www.ebXML.org). The goal of ebXML is to facilitate the creation of components and services, which can be used seamlessly within an integrated electronic global framework.

The ebXML model uses two models to describe relevant aspects of all business interactions, which are:

- **Business Operation View (BOV):**

This view is more data-centric in nature and addresses the semantics of business data transactions and associated data interchanges. This is specifically focused towards the business needs of ebXML partners.

- **Functional Service View (FOV):**

The Functional Service View focuses upon the supporting services and business need of trading partner involved in ebXML. This Function Service View undergoes under three different phases, which are implementation, discovery and deployment and the runtime phase.

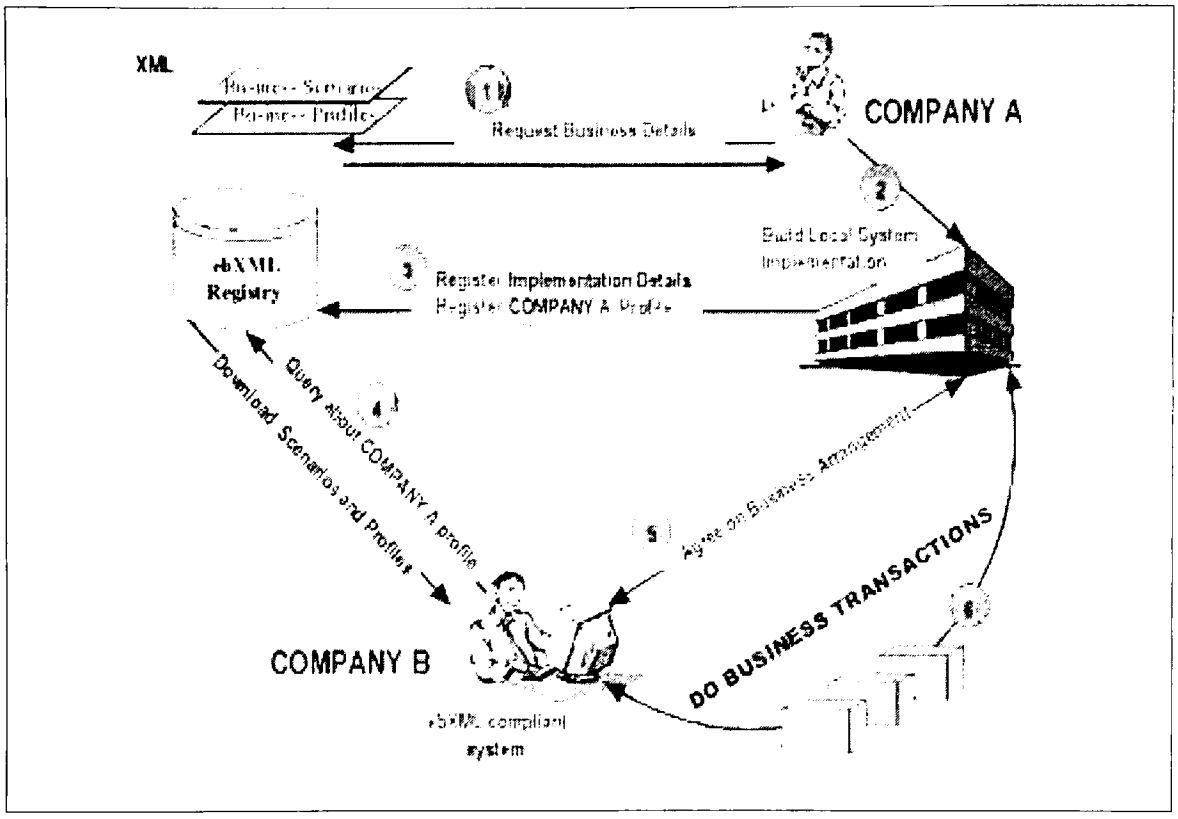

Figure 3.1, ebXML business-to-business interaction model (http://www.xml.org/feature_articles/2000_1107_dutton.shtml).

The ebXML architecture maintains a registry system to incorporate both BOV and FOV models. An ebXML Registry provides a set of distributed services that enable the sharing of information between interested parties for the purpose of enabling business process integration between such parties by utilizing the ebXML specifications. (http://www.xml.org/feature_articles/2000_1107_dutton.shtml). Figure 1, demonstrates an information sharing approach taken by two businesses where common information are maintained as object in the ebXML registry. The objects and information have to be created and maintained using the specifications provided by ebXML.org.

3.2.2. BizTalk.org

The goal of BizTalk.org is to provide resources for learning about and using Extensible Markup Language (XML) for Enterprise Application Integration (EAI) and B2B document exchange, both within the enterprise and over the Internet (http://www.biztalk.org). Biztalk is an initiative taken by Microsoft technology to use

"best of breed" technologies as XML and enhance business information interchange. "The BizTalk Framework aims to accelerate the adoption of XML with a set of design guidelines that encourage the consistent development of XML schemas, " said James Utzschneider, director of business frameworks for Microsoft (http://www.extensibility.com/company/headlines/biztalk.htm).

The goal of Biztalk.org is to provide a formal mechanism for exchanging XML interfaces. Biztalk.org is headed towards establishing a standard of set of vocabulary (template) so that data can be shared consistently among enterprises. For instance, all Accounting systems understand debits and credits, all Graphics systems understand lines and circles, and so on (Scribner and Stiver, 2000). The web site BizTalk.org hosts a repository, which enables enterprises and business to publish their standard schema. Organizations can publish their schemas by publishing registering through the bizTalk.org website. Enterprises using Biztalk.org to publish their xml schemas for information sharing have to comply with the Biztalk standard and specifications. Biztalk is an altemative to ebXML. However, Biztalk and ebXML can work in conjugation with each other in the business world.

3.3. Web Services

Implementing Biztalk or EbXML does not guarantee that businesses will find each other or document interchange will take place without any data loss. How do we write a schema repository, which is accessible by everyone? How do we implement services so that businesses can find each other without any difficulty? How do we make use of directories and services that already exist in the Internet? How do we enable business to advertise them on the Internet? It has slowly been realized that the universal information-sharing model provided by the Internet will be the solution to our problems. We need web services to do accomplish this. Web services are integration of Universal Description, Discovery and Integration (UDDI), which allows discovery of services on the web, Web Services Description language (WSDL) which is the language we use to create schema repositories and describe XML objects and Simple Object Access protocol (SOAP) which is the standard protocol we can use to send and receive XML packets between endpoints. We can discover, read vocabularies, build messages and transactions with a combination of these services. Using Web Services software can be available as services through the Internet and information can be shared at a very low cost and ease.

Web Services are self-contained, modular applications that can be described, published, located, and invoked over a network, generally, the Web (http://www-106.ibm.com/developerworks/webservices/library/w-ovr/). Web Services provides is an evolution of the object-oriented analysis and paradigm and follow the fundamental concepts of object-oriented methodology such as encapsulation, message passing, dynamic binding, and service description and querying. The goal of Web Services is to make components and legacy systems available as services with wrappers, which can be integrated at *run time* to produce dynamic and flexile application. Web Services can be considered to *run Time* components as opposed to *development- time* components we are accustomed to. Web Services are headed towards providing a loosely coupled

architecture in the e-commerce world as opposed to tightly coupled architecture provided by distributed architectures such as DCOM, CORBA and RMI.

IBM's Web Services Architecture (SOA-Service Oriented Architecture)

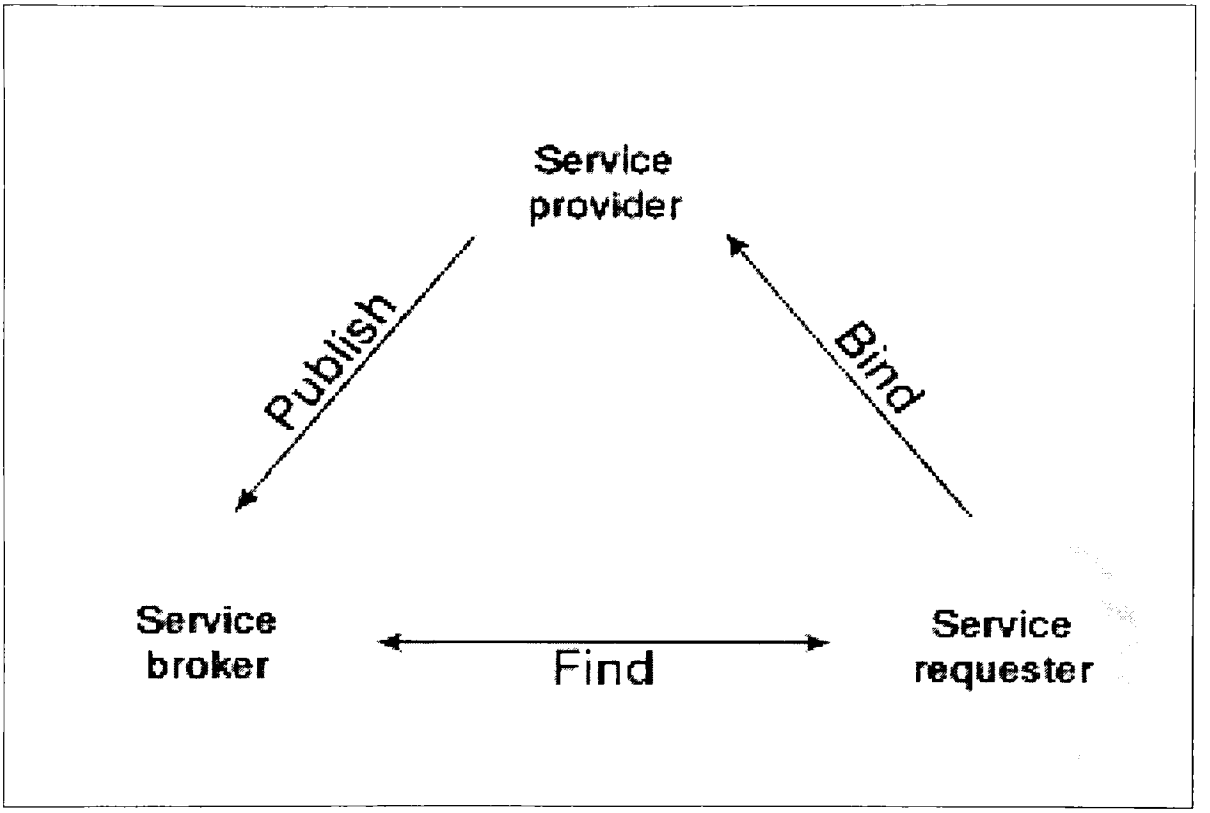

Figure 3.2, IBM's Service Oriented Architecture.

Different vendors have taken initiatives towards implementing Web Services architecture. Microsoft's .net implementation, which is work under progress at the time of this writing, focuses heavily into the Web Services Architecture. Figure 2 illustrates the Web Services architecture model purposed by IBM.

The IBM's Web Services architecture categories network components into three different groups as Service Provider, Service Broker and Service Requester. The Service Broker and providers hold a producer consumer relationship and use the Service Broker. The Service Broker acts as a central BUS between the supplier and consumers by maintaining a shared repository.

The three different operations of web Services according to IBM's architecture are:

- Publish

- Find and
- Bind

The Services Providers are required to publish their abilities to the Service Brokers. The publishing is carried out using XML-based Well-Defined Services (**WDS**) document. WDS document provides nonoperational information about services, which may be information and description of the service, service category, and the name of the company that created the service.

Service Requestors ask the Service providers to carryout the request through the repository maintained by the Service Broker. The Service Requester looks up known services based on the WDS document and the Network Accessible Service Specification Language (NASSL) document (http://www-106.ibm.com/developerworks/web/library/w-int.html). NASSL is an XML-based IDL, which describes the interfaces necessary to access the services provided by Service Providers.

3.5. Components of Web Services

Web services are an integration of SOAP, UDDI and WSDL. Web Services at this time is reaching the first stage of technology maturity. The software development world has slowly started to realize that web services are capable of accomplishing seamless interoperability, platform and language independence which many older and complex technologies were not capable of. Each and every major player in the software Industry is doing their best to take advantages of major web services enabling technologies as SOAP, UDDI and WSDL. However, web services enabling technologies have not yet beet tested on major enterprise level applications. This does not mean that Web services technologies will be a failure in achieving its goal in the software Industry. Enterprise ready tools such as Microsoft's .NET platform and the Apache Axis Engine (both of which are still currently in development) promise to deliver a high level of value, performance, and stability upon which to build existing web services (http://www-106.ibm.com/developerworks/library/ws-ref1.html).

3.5.1. Simple Object Access Protocol-SOAP

Simple object Access protocol is an xml-based protocol for the exchange of information in a decentralized distributed environment. SOAP can be considered to be a messaging protocol as well as an implementation. SOAP can be implemented into other web Services enabling technology. For example SOAP has been used in the implementation of Biztalk Framework where the routing, delivery, and transactions are all guaranteed. SOAP can be embedded within a web service enabling technologies as a set of tags. Ebxml.org has finally decided to use SOAP as a messaging protocol in their implementation.

SOAP is designed to work well with Internet where the systems are generally lightly coupled. Distributed systems as CORBA, DCOM and RMI are tightly coupled where enough information between the all the communication endpoints are required before the

implementation of the system. It is not possible to have a tightly coupled system in the Internet where most of the systems do not have the prior knowledge of each other. This is the reason a lightweight protocol as SOAP was required to work with the stateless nature of the Internet and build loosely coupled systems. Instead of browsing a web of linked documents and manually initiating requests by forms, SOAP allows us to access Web Services directly, by invoking method calls on remote objects using an XML message, and receiving an XML response document.
(http://www-106.ibm.com/developerworks/speakers/colan/).

3.5.2. Universal Description, Discovery and Integration (UDDI)

The Universal Description and Integration is an open initiative taken by the industry to enable enterprises to discover each other and define how they interact with each other sharing information in a global architecture. UDDI is the building block, which will enable businesses to quickly, easily and dynamically find and transact with one another via their preferred applications (http://www.uddi.org). UDDI is platform independent business initiative which enables enterprises to discover describe and integrate business services using the Internet. UDDI is a global business originally initiated by business and technology leaders as Arbia, NTT, Nortel, IBM, and Microsoft. UDDI is not owned by any single Industry.

The UDDI specification is not industry specific and was initiated for the benefit of business of any size. Previously, there was no universally accepted mechanism for enterprises to reach their customers and trading partners with information about their web services and products. UDDI enables business to be discovered easily in the Internet. This also enables organizations to discover the right business match easily from numerous businesses which are available online.

The UDDI specification was designed with the motive of taking advantage of the various Internet specifications provided by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) standards. UDDI takes advantages of the latest IETF standards as Extensible Markup Language, Hypertext Transport Protocol, Domain Name System (DNS) protocol and Simple Object Access Protocol in its initiative to discover businesses online by the implementation of Web Services. Businesses are required to register online and provide information regarding their products and business perspective. The information provided by the businesses is stored according to the UDDI specification and available in searches to potential buyers and trading partners.

There are numerous business directories like UDDI, which already exist in the business world. This directory lists basic information on an enterprise as name, address, contact and phone numbers. However, they do not list the programmatic descriptions of Web services businesses have to offer. UDDI was initiated with the motive of provide a centralized source of information for already existing business directories. With the help of UDDI it is easy to find out the details on technology and application used by the businesses. For example, using UDDI businesses can find out trading partners who use the same technology as they do.

UDDI is similar to the initiatives taken by ebXML and Biztalk. The fundamental difference between UDDI and ebXML is that UDDI is aiming to create a standard registry for companies that will accelerate the integration of systems round Net Marketplaces, while ebXML is working to standardize how XML is used in general business-to-business (B2B) integration (http://www.xml.org/feature_articles). UDDI is geared towards acting as a middleware technology to describe systems that enterprises used to interact with each other using XML.

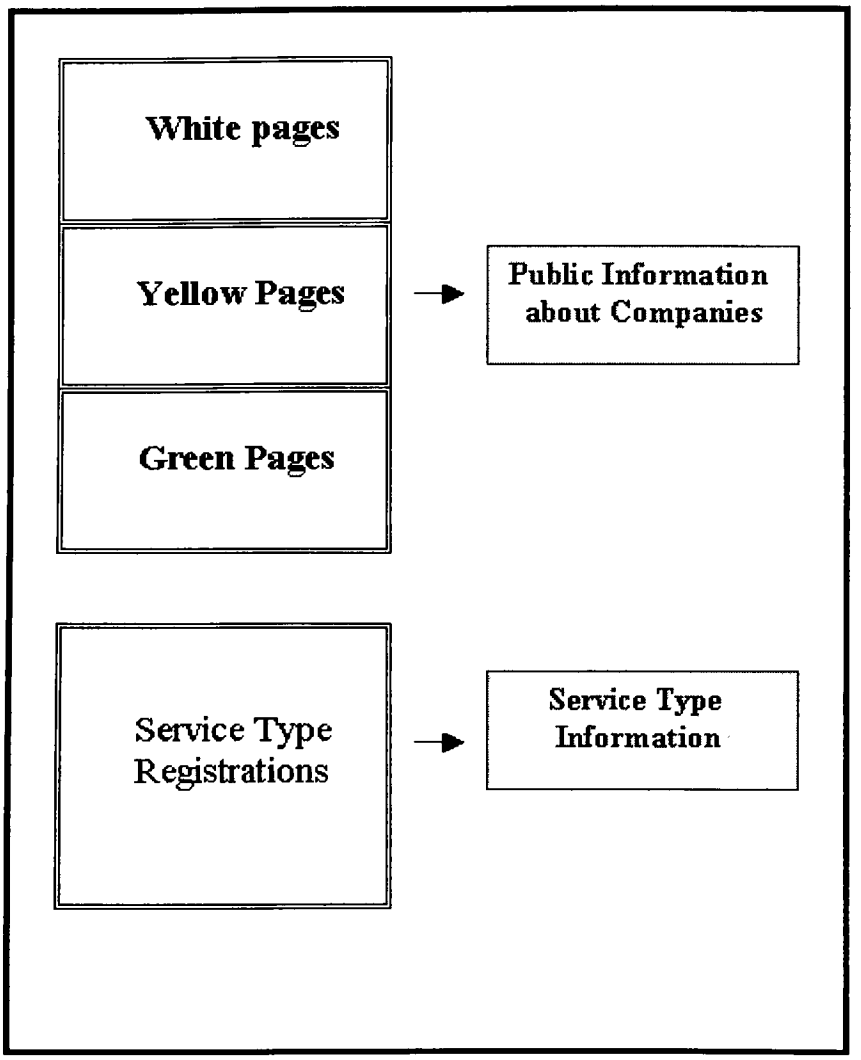

Figure 3.3, UDDI directory structure layout

UDDI plans to provide company information and profile in a shared directory accessible freely on the Internet. Existing ebXML, Biztalk, EDI systems or any other directory services can be described in the UDDI directory. UDDI directory is based on XML but the registry can describe services dependent on CORBA, DCOM or any other distributed System technology.

The UDDI business directly categorizes enterprise information into three groups as (figure 3.3):

White pages:

The white pages directory enable companies to register information as name and the service provided.

Yellow Pages:

The yellow pages component of the UDDI directory will categorize companies according to NAICS industry standards codes set by the U.S. government, by United Nations/SPSC codes and by geographical location.

Green Pages:

Green Pages elements are the final elements of the UDDI directory. Using Green pages companies can interact with other companies in the registry using XML.

The bottom layer of UDDI directory layout consists of the service description. This is where standard body of developers and organizations register their information about the services provided.

3.5.3. Web Services Description Language (WSDL)

Web Services Description Languages, is an XML-based grammar for describing the location, functions, properties and the invocation of Web Services. The Web Services Description Language is the XML equivalent of a resume -- it describes what a Web service can do, where it resides, and how to invoke it (http://www-106.ibm.com/developerworks/webservices/library/ws-peer4/). One of the major reasons behind implementing the Web Services architecture is to have the ability to develop applications dynamically by discovering components available as services in the network. **WSDL** is the resulting artifact of a convergence of activity between NASSL (IBM) and SDL(Microsoft)(http://www-106.ibm.com/developworks/webservices/library/ws-arcl/). WSDL is designed to provide a platform and protocol independent way of describing the Request/Response model for invocating remote methods that are available as services in the Internet.

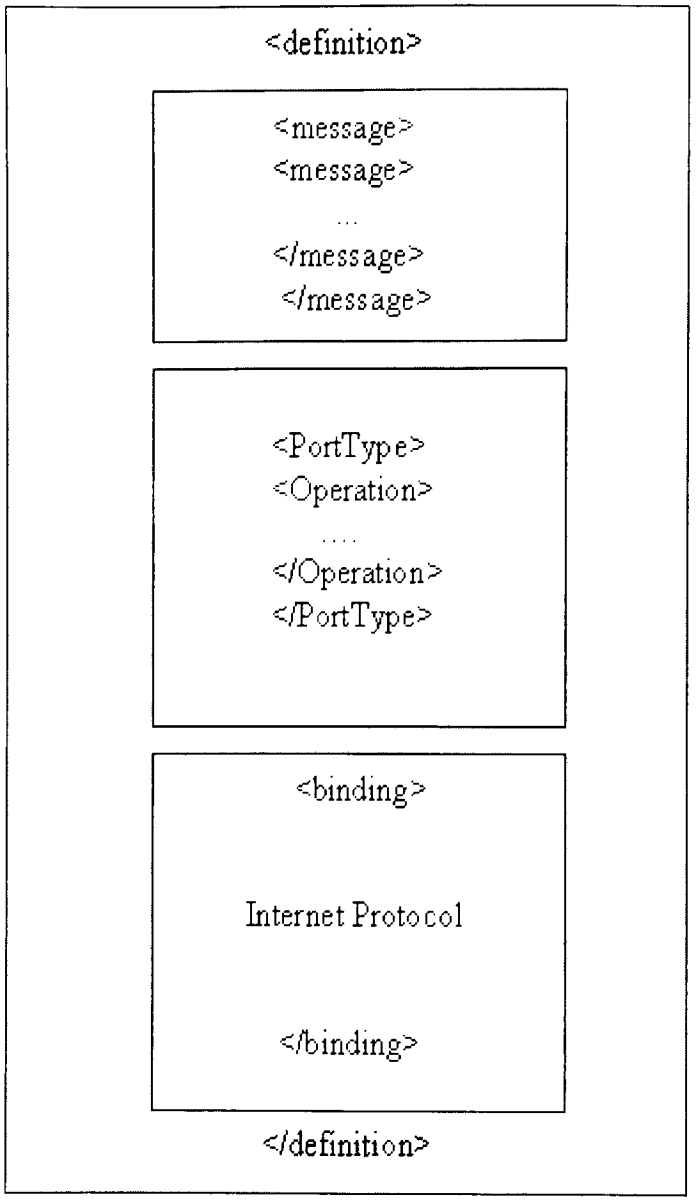

Figure 3.4, WSDL document Structure

Figure 3.4, illustrates the standard structural of a WSDL document used in describing Web Services. The standard XML elements used in describing WSDL documents are:

Definitions: <Definitions>

The definition is the root element of the WSDL document. The definition element holds the definition of one or more available services. The lists of attributes, which may be used in describing the definitions elements, are:

- **name:** This attribute is used in describing the overall purpose of the services and is optional.

- **tatgetNameSpace:** This attribute is used in defining the logical namespace on for the information about the service.

- **Xmlns:soap and xmlns:xsd: This** attribute is used in defining any SOAP or XML messaging architecture used.

- **Xmlns:** This is the attribute that describes the default namespace for any WSDL document. The default namespace for a WSDL document is http://schemas.xmlsoap.org/wsdl.

Message: <message>

The message element is contained inside the definition element. The message element is a single piece of information shared between the invoker and the service. A regular round trip has to be modeled as two messages according to the WSDL specification.

PortType: <PortType>

The PortType element is contained inside the definitions element and corresponds to a set of one or more operations. Each operation defines a particular instance of input/output message sequence.

Binding: <binding>

A Binding element is contained inside the definition element and defines the type of Internet protocol used in the communication. WSDL is designed to be protocol neutral and bindings can be specified for SOAP, CORBA, DCOM and any other standard Internet protocol.

Service: <Service>

A service in a WSDL document is used to represent a collection of ports. Each port is an element in a WSDL document and represents the availability of a particular binding at specific endpoint.

Documentation: <documentation>

The documentation element is an optional element in a WSDL document and contains human readable description of the element. This is used for maintenance and debugging purpose.

3.6. B2B in General

Web Services are Integration of HTTP, SOAP, UDDI and WSDL. The distributed nature of the Internet provides enough room for sharing information across between enterprises in the business world. The major architects of Web Services are headed towards providing software as services using the Internet. Business can register and request services through the UDDI central repository using SOAP messaging architecture. This enables dynamic discovery and definition of businesses in the global market place. This will also help in bridging the gap between businesses of different sizes and at different locations. All of these are based on XML which means work services can work well with different platforms, languages and environments.

If we look into the XML world recently we can see that so many standards are emerging everyday. The Internet is going under its evolutionary phase. The Evolution of Web Services can be compared to the Darwin's Theory of Evolution of Life. During the Evolution of life multi-cellular organisms emerged from single cellular amoebas. During the evolution of multi-cellular organisms, the evolution phase went across various means of communication between the cells. The most favored mechanism of communication between the cells survived. Similarly, computing is also evolving from single units to a wide distributed architecture. XML is another undergoing phase in the evolution of distributed computing. All the standards and methods emerging with Web Services are part of the Evolutionary Phase. Like in the Theory of Evolution of Life, the most dynamic and efficient methods and standards will be accepted with time to achieve pure distributed systems architecture. This is where the Web Services evolution is headed into – resulting effect of XML, SOAP, UDDI and WSDL.

References

"Biztalk Community", Microsoft Corp. < **http://biztalk.org/home/default.asp**>

"What is EbXML?", EbXML,<**http://www.ebxml.org**>.

"Energize e-business with Web services from the IBM WebSphere software platform". IBM. <http://www-106.ibm.com/developerworks/webservices/library/ibm-lunar.html?dwzone=webservices>

"Making the Network for B2B", Ariba. <http://www.ariba.com>

"Universal Description, Discovery and Integration" UDDI.ORG. <http://www.uddi.com">

"Web Services Description Language (WSDL) 1.0" Developerworks. <http://www-106.ibm.com/developerworks/webservices/library/w-wsdl.html?dwzone=webservices>

"Web Services and UDDI overview". IBM. < **http://www-3.ibm.com/services/uddi/**>.

"Web Services Description Language (WSDL) 1.1", Microsoft Corporation. <**http://msdn.microsoft.com/xml/general/wsdl.asp**>

Burbeck, Steve. 'The Tao of e-business services'. <**http://www-106.ibm.com/developerworks/webservices/library/ws-tao/index.html?dwzone=webservices**>

Fisco, Dave. "IBM's Web Services architecture debuts" <http://www106.ibm.com/developerworks/web/library/w-int.html> (September 2000).

Glass, Graham. "The Web services (r)evolution" <**http://www-106.ibm.com/developerworks/webservices/library/ws-peer1.html?dwzone=webservices**>

Hertzfeld, Andy. 'Easier System Management' <**http://www.futureofsoftware.net/ah0010/ah0010.asp**>

Walker Whitlock, Natalie. "The missing link Will ebXML bring true global B2B?". < **http://www-106.ibm.com/developerworks/webservices/library/ws-ebxml.html?dwzone=webservices**>.

Smith, Rod A. E-business Software. < **http://www.futureofsoftware.net/rs2_0010/rs0010.asp**>.

4. eXtensible Markup Language

4.1. Electronic Document Interchange

Electronic Document Interchange is the direct communication of trading messages between computer systems, using national and international telecommunications networks. (http://www.bic.org.uk/infopak.html#edi). Electronic data Interchange has existed since the past twenty years. It did not just evolve with the growth of Electronic Commerce as many of us may be thinking. Exchange of documents and information has always played a prominent part in the business world. In the past companies have achieved exchange of information through mailing and use of preprinted business. With the growth in the use of computers companies decided to shift towards using computers in the exchange of documents. The exchange of interchange Documents electronically reduced paperwork, human interaction, increased response time and reduced cost. EDI was simply a method of sending text streams across value added network.

EDI trading messages can be as basic as orders and invoices, but EDI can also develop into a much more sophisticated information exchange, so that trading partners manage their whole supply chain more effectively (http://www.bic.org.uk/infopak.html#edi). Traditionally EDI was used simply to send and receive information electronically. To use EDI to exchange information, EDI message handling software had to be running on both end. In the earlier days companies used their own format to transfer EDI messages from one point to another. Due to the growth in the number of companies and business partners a standard format for transferring EDI message was required. It was not only till 1970 work was done towards establishing a EDI standard in exchanging information. Many standards have been developed so far but the most common and widely used standard includes the X12 and the Electronic Interchange for Administration, Commerce, and transport (EDIFACT). The X12 was initiated by the American National Standard Institute to facilitate the electronic exchange of business information. Similarly, EDIFACT is a standard devised by the International Organization for Standardization. Proprietary software is needed to construct a EDI messages and then load them into Advantis, Gentran or Harbinger for transmission over a private leased line. A typical EDI message sent across the wire might have looked like in Figure 4.1 below:

```
1999 123FGS EMERSON OLDED PRODUCTS PO1227663
12 INJ MOLDED FRAMMISES BESTWAY 199 EA
0201200002032000 1030
```

Figure 4.1. Sample EDI Message.

The EDI message contained the length of each field and structure of the one to many relationship maintained by the fields before being transmitted over the wire.

Problems Encountered by Electronic Document Interchange

EDI has been accepted and used by industries over more than twenty years. This does not mean that it has no problems associated with it.

Cost:

EDI is very expensive for midsize and smaller enterprises to use. This is one of the reasons EDI has come out to be popular with only the top fortune 100 companies. The software needed to construct and read EDI message are costly and difficult to maintain and keep up with the changing standard.

Problem of Mapping:

Enterprises have the advantage of storing internal information into Relational databases. However, the notation used by EDI messaging standard does not seem to follow a structured pattern. This makes it complicated to map relational databases with standard EDI messages. Software, which is required to do the mapping, tend to be very expensive especially for small sized enterprises.

EDI on the Internet

Due to the high cost associated with doing EDI using Value Added Networks enterprises started looking towards using the Internet. Internet mail was used to transmit EDI message over the Internet. Due to the security issue and unreliability of the mail servers this practice became less popular.

With the advancement in the browser technology, the web started being used as a means of transmitting EDI message. The larger company of the supplier provides web forms to the clients. The clients or the consumer submit the web forms to the suppliers web server. The form submitted is processed on the service using technology such as CGI, ASP, Pearl, Java Servlets or Active Server Pages.

XML/EDI

XML, with its ability to transfer structured data over the Web, immediately attracted the attention of people and organization struggling with traditional EDI (http://www.xml.com/pub/a/1999/08/edi/index3.html). It seemed so obvious when someone could easily wrap tags around the EDI components. Figure 4.2 below shows the same information as in Figure 4.1 but with tags wrapped around the data.

```
<year>1999</year>
<transactionID>0123FGS</transcationID>
<companyname>EMERSON MOLDED PRODUCTS</companyname>
<transactiontype>PO</transactiontype>
<ponumber>1227663</ponumber>
<quanitity>12</quantity>
<product>INJ MOLDED FRAMMISES</product>
<ship>BESTWAY</ship>
<price>199</price>
<unit>EA</unit>
<earlyarrivedate>02/01/2000</earlyarrivedate>
<latearrivedate>02/03/200</latearrivedate>
<discountpercent>10</discountpercent>
<termdays>30</termdays>
```

Figure 4.2: Data with tags around them.

XML facilitated the use of these structured and user defined tags, which are also extensible in nature. XML has the ability to fill the gap in-between small enterprises and larger enterprises in the Electronic Exchange of Documents. XML tags are objects and they can be used as containers to hold the EDI documents or any other objects. The business model provided by the web is fairly affordable even by small enterprises and it is easier and less expensive to maintain. XML related software is available free of cost or at a very reasonable price. XML fills the gap in-between the various standards that vary between different countries and nations. XML seems to solve the problem of mapping the enterprises internal data storage system and messaging documents. XML provides more ways of displaying and processing data using Extensible Style Sheets (XSL). XSL enables visual display and formatting of incoming data. Using XSL, enterprises can convert the incoming data directly into formats they want. This is where the role of XML comes into play. XML can be easily mapped into Enterprises internal data storage, XML is affordable and easily consumed by web services. This is the reason XML started becoming ubiquitous even though it started in the late nineties. XML emerged as a replacement for EDI and its popularity is growing exponentially with even more promises.

4.2. History Of Markup Languages

We have come across HTML, which stands for Hypertext Markup Language, XML, which stands for Extensible Markup Language, and SGML, which stands for Standardized General Markup Language. Markup languages evolved from simple Procedural Markup language to Generalized Markup Language through generic coding. The idea of implementing Markup Language emerged from the Publishing Industry.

Markup is a separate activity that takes place after writing and before typesetting (Marchal, 1999)

Evolution Of Markup Language

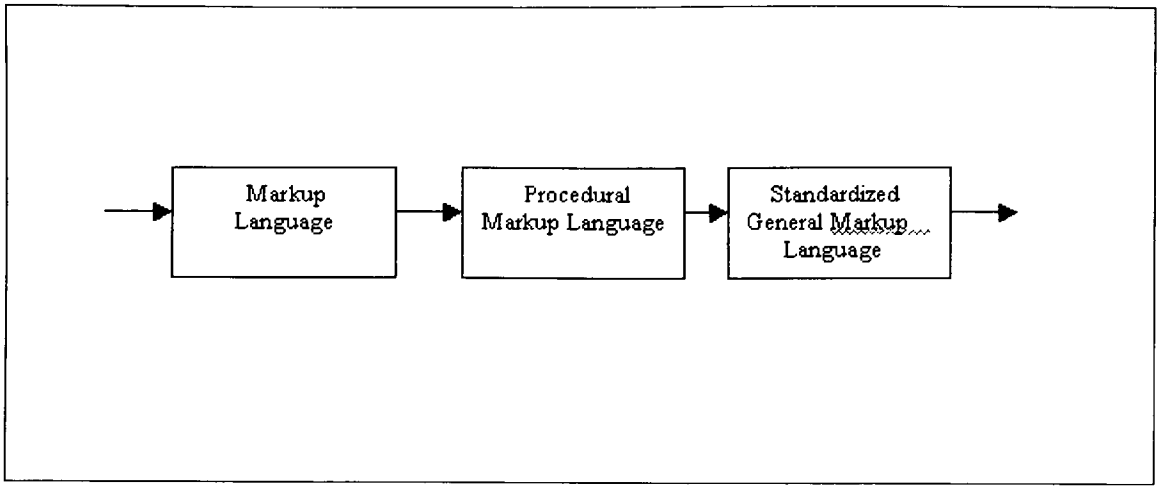

Figure 4.3: Evolution of Markup Languages

During word processing the user is required to manipulate the appearance of the text that is going to be printed. The user can choose to out various styles into the text such as paragraphing, indenting, size and color. The user has a list of menus that can be used to style the written text. In this case the user determines the set of commands, which are Required to format the appearances of his text. Since, the Markup used in this context seems to be set of instructions or procedures for some output devices it is called Procedural Markup Language. However, Procedural Markup Language seems to be a slow process and also, it does not provide a way to store the structure of the document.

Generic Coding

Gradually, Generic coding evolved from the emergence of macros into computing. Macros can be used to make calls to format from external sources. The use of general coding provides more flexibility as the macros can be changed instead of having to redo the whole document. The progression of Markup into General encoding also takes it closer to describing the structure of the document as a whole.

Standard Generalized Markup Language

SGML or Standard Generalized Markup Language extends the functionality of General Coding. SGML was developed upon the standard defined by Dr. Charles Goldfarb. He invented the concepts behind SGML. SGML represents the structure of the document in contrast the to Procedural Markup Language and General encoding which focuses on the appearance of the text. SGML conforms to a model in terms of storing data similar in context to database schemas. Software can be designed to process SGML data. SGML

can also be processed to fit into relational database and converse. SGML does not define a set of tags as Generic Coding or Procedural Markup Language does but it imposes the flexibility for author to define his/her own tags and the structure of his/her own documents. SGML resembles to procedural programming language since it is a enabling standard to describe structure of documents.

4.3. Extensible Markup Language-XML

XML or Extensible Markup is a markup language created to design documents and to store structured information. Structured documents can contain any form of data as text, pictures, sound etc. A markup language is a mechanism to identify structures in a document (http://www.xml.com/pub/a/98/10/guide1.html#AEN58).

XML is a metalanguage, which was inherited from Standard Generalized Markup Language (SGML). A metalanguage is a language used to describe other language. An alternative to SGML had to be created because it did not seem to be work very well when it came to storing and transmitting documents over the World Wide Web. SGML was also too difficult to be implemented for the web browser. A very high cost was also associated with implementing SGML to be used with the WWW.

The architect of the World Wide Web Tim Berners-Lee defined limited number of tags, which would make it easier to develop browsers for all documents transmitted over the web. This set of predefined tags came to be known as Hypertext markup Language or **HTML**, which came to be used extensively with the browsers in displaying web pages to the users. However, Tim Berners-Lee only focused upon how documents were presented and not how the documents were stored and processed over the World Wide Web. However, HTML was good only to be used in displaying human readable documents on the browser. HTML did not provide any structured tags for software systems to process it.

The number of Web users grew exponentially but did not have the capability for processing information. People started realizing the fact that the only available alternatives SGML and HTML did not seem feasible when it came to sharing and processing information over the World Wide Web. This is why the W3C Consortium decided to come up with Extensible markup language, which was to be inherited, from Standard Generalized Markup Language. XML was finally developed in 1997 by the W3C Consortium and has been accepted widely today. XML gained popularity because of its simplicity to be used over the World Wide Web. XML also proved to be successful when it was used as a means to transmit electronic document among enterprises. This has also emerged to be an alternative for Electronic Document Interchange standards especially among mid-sized and small-sized enterprises.

The Application Of Extensible Markup Language

XML has not only been used for storing structured documents but also for displaying the document to the web browser, which is viewable to the user. XML can be used in document applications where information can be manipulated and displayed to. XML documents can be displayed into various human viewable formats as HTML, PostScript, RTF and more. XML documents are converted automatically into the specific format using style sheets. XML can also be used in exchanging data in-between enterprises. XML has already started replacing Electronic Document Interchange because of its affordability and compatibility with the web.

4.3.1. Components of XML documents

XML documents are made up of markup and document. The different kind of markup associated with an XML document includes xml declarations, elements, attributes, reserved attributes, entity references, comments, Character data sections, processing instructions and document type declaration.

```
<?xml version="1.0"?>
<Products>

      <Product ID = "1000">
            <ProductName>Norton Bolts</ProductName>
            <Units>2</Units>
            <Price>33</Price>
      </Product>

      <Product ID= "1001">
            <ProductName>EMG Conductor</ProductName>
            <Units>5</Units>
            <Price>1500</Price>
      </Product>

      <Product ID="1002">
            <ProductName>Sony Radio</ProductName>
            <Units>2</Units>
            <Price>150</Price>
      </Product>

      <Product ID="1003">
            <ProductName>Micorsoft .net</ProductName>
            <Units>4</Units>
            <Price>249</Price>
      </Product>
</Products>
```

Figure 4.4: XML code sample for products.xml

XML structure is called a *tree structure*. The topmost element of an xml document is called the root and the succeeding elements are called child elements. Each child element is eligible to have a sibling, child, or parent node element. Figure 4.5 shows a tree structure for the xml document products.xml as show in figure 4.4. The root element is the element Products and rest are the child objects.

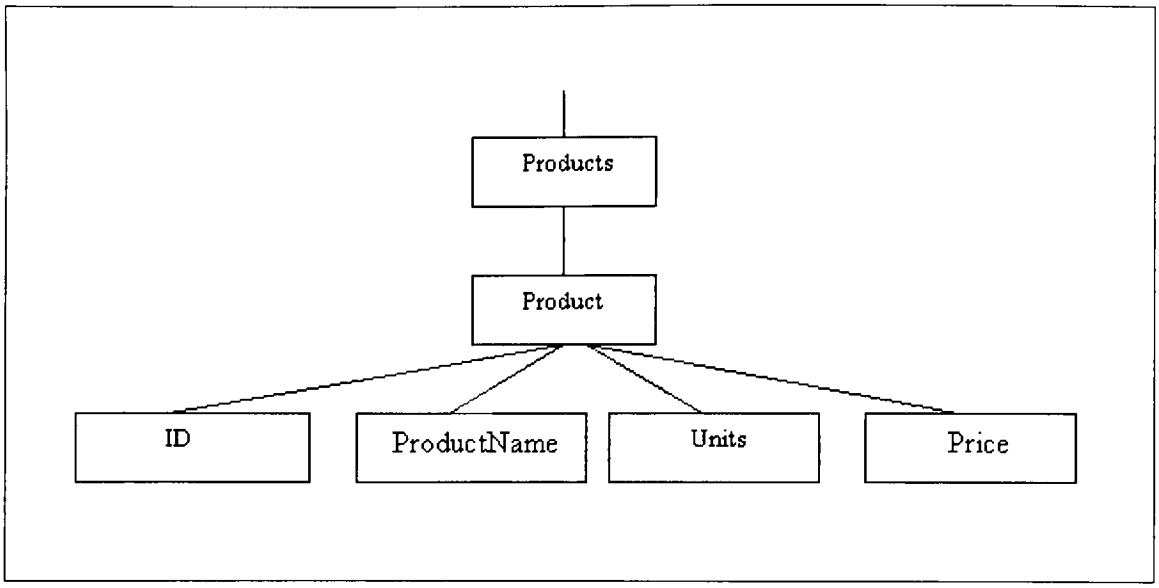

Figure 4.5: Xml tree structure for the file products.xml

4.3.1.1. XML Declarations

An xml document should start with a special tag called **xml declaration**. This xml declaration can contain up to three pieces of information. If we refer to Figure 4 we can see that the xml document products.xml starts with the tag

<? xml version ="1.0"?>

This specifies the **version** number of xml specification used. The word xml is case-sensitive and has to be strictly lower-case. The xml declaration can have an optional tag **encoding** which will make it

<? xml version = "1.0" encoding="UTF-8"? >

The character encoding tag identifies the character encoding schema used in the xml document.

The xml declaration tag can also have an optional **standalone** tag, which will make the above document appear

<? xml version = "1.0" encoding="UTF-8" standalone="Yes"? >

The standalone tag reveals if an externally defined set of declarations contain information that affect the processing of the xml document.

4.3.1.2. XML Elements

Elements are the most widely used markup in an xml document. An element, which contains data, begins with a **start-tag** and ends with an **end-tag**. Our code example in figure 4.4 contains an element called ProductName. This element starts with a tag <ProductName> encloses a data in-between and ends with a tag </ProductName>. If the element <ProductName> does not contain any data the syntax <ProductName/> will be sufficient to describe the element.

An element itself can be considered as an object and can be used to hold other objects. An element can act as a container element to hold other elements. For example, Product element acts as a container element, which is used to hold other elements as ProductName, Units and Price.

4.3.1.3. XML Attributes

An attribute tag in xml document is used to hold additional information on an element. Attributes have to be embedded into the element start tag. In our xml document from figure 4.4 products.xml

<Product ID = "100">

ID is an attribute, which holds the information on identification number of the product. The value associated with the attribute has to be enclosed in quotes. There has to be at least one space, which separates the element and the attribute.

4.3.1.4. XML Reserved attributes

Reserved attributes are some universal characters, which may be shared by more than one application. These are used to avoid conflict in-between user defined tags and universal tags. The xml standard contains only two reserved attributes.

4.3.1.5. XML Entity references

Some of the characters have been reserved to identify the starting of the markup in an xml document. For example the character, <, has been reserved to start the beginning of a tag. To insert the character, <, into our xml document it has to be represented in a different notation. Entities are used in xml to represent these special characters. Each entity in a document has to be identified with a unique name. For example if we want to use the character, < in our document then we have to represent it using <. In xml entity reference begin with the ampersand and end up with a semicolon.

4.3.1.6. XML Comments

It is possible to put comments within xml document as in other programming languages. This is a concept inherited from other Markup languages such as HTML and SGML. An xml comment section may be defined as

<!- -This is a comment - ->

Comments help in organizing documents and make it easier to maintain and follow.

4.3.1.7. XML Character Data Sections

Character data sections are used to identify a block of text, which will not contain a Markup. Markup characters are not expected in Character Data sections. Scripts can be embedded into xml documents using character data sections.

```
<xsl: script>

<![CDATA[

        function lineTotal(item)
           {
            //this function will calculate the Unit Total
            var price = item.selectSingleNode("Price").nodeTypedValue;
            var qty = item.selectSingleNode("Units").nodeTypedValue;
            return qty*price;
           }

        ]]>

</xsl: script>
```

The above is a portion of xml Style sheet that uses character Data Section to embed Java Script code into it. The Character data Section begin with the tag **<![CDATA** and end up with the tag]]>.

4.3.1.8. Processing Instructions

Processing Instructions or (**Pis**) are escape hatch to provide information to an application (htttp://www.xml.com/lpt/a/98/10/guide2.html). From our example xml code in figure 4.4 we have the code

<? xml version ="1.0"?>

where ? is the processing instruction. Processing instructions are similar to comment sections on xml documents, which are not actually a part of the xml document. Processing instruction can be used to identify the information on an xml document. Processing instruction can also be used to identify a style sheet. Style sheets are used to format the xml document into user readable format. For example the following text

<?xml-stylesheet type="text/xsl" href="summariseorder.xsl"?>

embedded inside a processing instruction references a style sheet summariseorder.xsl, which is used in displaying the xml document to the browser.

4.3.1.9. Document Type Declaration

One of the reason xml has become so popular is because of its flexibility to create user defined tags. Document Type Declaration is an optional feature in xml documents. But at some places it might be beneficial to use DTD (Document type Definitions). If we are using default value inside xml documents them it might be a good idea to declare out default values using Document Type Declaration. DTD are also used when white spaces have to be identified in xml documents. DTD are also equally important in applications where there is an interaction with Enterprises relational databases. DTD can be defined inside an xml document or a reference can be made to an external DTD file from an xml document. External files containing Document Type Declaration usually have a .dtd extension. The declarations, which are internal to the document, are referred to as **internal subset** and the declaration, which are external to the xml document, are called as **external subset**.

There are four kinds of document type declaration in xml, which are:

1) Element Type Declarations
2) Attribute List Declarations
3) Entity Declarations and
4) Notation Declarations.

Element, attributes and Notations comprise the physical structure of an xml document. The entity Declaration signifies the physical structure of an xml document.

4.3.1.9.1. Element Type Declarations

The keyword '**Element**' in a document Type Declaration identifies an element declaration. An element in an xml document has to begin with a letter, an underscore character or a colon. It is not mandatory for an xml element to have content. An empty element can be denoted by the keyword **EMPTY**. An element may or may not have child elements. The declaration for an element containing child can contain either the keyword ANY or a model group.

Model Group can be used to describe other enclosed elements or text in xml Document Type Declaration.

```
<! - - this is a the document type declaration for products.xml

<!ELEMENT      Products          (Product*, ProductName*, Units*, Price*)
<!ATTLIST      Product  ID  id    #REQUIRED
<!ELEMENT      ProductName       (#PCDATA)
<!ELEMENT      Units             (#PCDATA)
<!ELEMENT      Price             (#PCDATA)
```

Figure 4.6: DTD sample file for products.xml

A model group has to be bounded by parenthesis and contain at least one **token** that may be name of an element contained. In figure 4.6, in the declaration

<!ELEMENT Products (Product*, ProductName*, Units*, Price*)

Elements <Products>, <Product>, <Units> and <Price> are defined to be the model group. A model group can contain more than one token. The contents inside the model group can be organized using special characters as +, *, or |.

The sequence or the order of the items inside the model group can controlled by using the, or the | character. For example, the model group (Product, Productname, Units) indicates that the element has to be defined in same order starting with Product and ending with Units. However, if the model group is defined as (Product | ProductName | Units) the elements can be declared in any order.

The designed of DTD has the privilege of indicate the number of times the element can appear at each location. The following characters are used to control the quantity of elements inside an xml document:

1) A '+' beside an element declaration indicates that the element should appear several times in the document

2) A '*' beside an element declaration indicates an element should appear zero or more time in the document

3) A '?' beside an element declaration indicates an element does not have to appear in the document.

The keyword **#PCDATA** in dtd section indicates that the element contains a text.

4.3.1.9.2. Attribute List Declarations

Attributes List Declaration is used to identify which elements may contain attributes. Attributes hold additional information on an element. In our Document Type Declaration in Figure 4.6, we have

<! ATTLIST Product id ID id #REQUIRED

which is used in declaring an attribute id for the element named Product. The tag #REQUIRED indicates that it is mandatory to have an id value for the element product. Attributes can be given any name but the same name cannot be used again within the same element.

In an attribute List declaration the first parameter identifies the name of the element and the second parameter describes the attribute type. So, attribute type for the id element in product will be ID. Attributes can be defined to be the following type during a document type declaration and are:

1) CDATA: The CDATA attributes are designed to contain strings of characters.

2) ID: The ID attributes uniquely identify an individual element in a document. These are similar to primary keys in database tables. The ID attributes are entitled to have a name as value.

3) IDREF or IDREFS: IDREFS are similar to ID but they have to be the value of some single ID attribute in the document.

4) ENTITIES or ENTITY: The entity type in a declaration is a special scenario and it indicates that the attribute value is an entity reference.

5) NMTOKENS: NMTOKENS attributes are similar to CDATA's and can contain only single word.

6) A list of names: These are similar to the enumerated data type in procedural programming languages like C and C++. The value of a particular attributes can be taken from a list of names.

The final parameter in an attribute list declaration specifies a default value. This is useful when the document designer does not enter a value. There are four different values, which can be associated with an attribute declaration and are:

1) #REQUIRED: This indicates that it is mandatory for the attribute to have a value in the document.

2) #IMPLIED: This indicates that the attribute does not have to contain a value necessarily. The processor can validate the document without the attribute value.

3) #VALUE: This indicates that a particular value can be assigned as a default value.

4) #FIXED: This indicates that the attribute defined does not have to have a value. If the attribute has a value then it has to have the #FIXED value.

4.3.1.9.3. Entity Declarations

XML documents can be distributed among a number of different files. An entity facilitates sharing of contents between various xml documents and can be considered to be physical units of storage. Entities enable us to include non-xml data in xml documents. For example, we can store a picture inside xml document by declaring it as a binary entity. Each entity has to be identified by a unique name. However, a document entity does not have to be assigned an entity name. Document entity is the data file, which is used in storing the entire document. Entities are considered to be the physical structure of xml documents. When the entity content is stored inside the xml document then it is referred to as internal entity. When the entity content is declared outside an xml document then we call it external entity.

Entities need to be declared inside the document. There are six different kinds of entities used in xml:

Internal text entities:

Internal text entities declare a name with a sequence of literals. For example, the declaration

<! ENTITY AUTHOR "John Doe" >

declares AUTHOR as an internal entity, which holds the string *John Doe.* A reference has to be made within the document to access the value represented by this entity. Using **&AUTHOR;** within the document we will be able to access the value represented by the entity which is **"John Doe".**

Built-in Entities:

Built-in Entities are similar to internal entities but are predefined by xml specification. There are five different kinds of Built-in Entities, which are:

- < declares the left angle bracket
- > declares the right angle bracket
- & declares the ampersand
- ' declares a single quote character
- & quote; declares a double quote character

Character Entities:

Character Entities are used to reference the extended ASCII character set, ISO 8859/1 and the Unicode/ISO ISO106466 in an xml document. For example '&#60' refers to the left angle bracket symbol in when declared as a character entity. A character entity reference is similar to internal text reference except that it uses a hash symbol after the ampersand.

Parameter Entities:

A parameter entity is declared using a percent sign '%' after the 'ENTITY;' keyword. The percentage sign instead of the ampersand is used in referencing a parameter entity within an xml document. For example in the declaration shown below, the symbol %code; references the parameter list, which is of the CDATA type.

<! ELEMENT %code (#PCDATAl list)* >

External Entities:

External entities are used in accessing information in another file or document. External entities declarations allow xml documents to acquire information from another file. External entities can be used to access both text and binary data. When external entities contain text, the content is inserted when it is reference and parsed. However, when an external entity contains binary data it cannot be parsed. When an External entity contains Binary entity it can only be referenced in an attribute.

External Entities can be represented by either **public** or **system** identifier. If we used the system identifier to provide the location of the location of an external entity, we have to conform to the URL standard. The following declaration

<! ENTITY mydatas SYSTEM "/ENTS/mydata.xml" >

uses the system identifier to an external file mydata.xml. Then the contents of the file in the file mydata.xml can be referenced using &mydatas; from within the xml document. However, using public identifier instead of system identifier provides more information on the content of the data file and also does not directly specify the location of the data file.

Binary Entities:

Binary Entities are similar to External entities but are strictly used to store pictures. They use the same concept of public and system identifiers as in External entities. However, in declaring a binary entity we have to add some more information regarding the type of the file used. For example to declare a picture file mypicture.tif we have to use

<! ENTITY myphoto SYSTEM "/pics/myphoto.tif" NDATA TIFF>

where NDATA indicates that picture is stored in TIFF format. The Binary entity has to be referenced using the call <pic name = "myphoto"/> within an xml document.

4.3.1.9.4. Notation Declarations

Notation Declarations are used when an element or entity contain non-xml compliant data. The declaration has to specify the type of file format to be embedded into the document. The keyword **NOTATION** identifies a notation declaration and has to be followed by a notation name. For example, we can declare a file runcommad.exe in an xml document by using

```
<! NOTATION        runCOMMAND        SYSTEM
        "C:\system32\runCOMMAND.exe">
```

The file runCOMMAND.exe can now be included inside the xml document.

4.3.2. SDL – Schema Definition Language

Document Type Declaration is a concept inherited from SGML. DTD's provide an excellent way of reusing existing definition and standard. However, the major drawback faced by DTD is that it only supports two different kinds of data types CDATA and PCDATA. Another bigger drawback is that DTD's are not well-formed xml document themselves. DTD's also do not support the object-oriented paradigm supporting only two data types and having no namespace support.

SDL (Schema Definition Language) were purposed to have support for more data types and conform to the object-oriented standard. Developers who are used to working with object oriented programming constructs find XML schemas to be much more appealing than DTD. However we should not forget that schemas are still work in progress and will still take few years to be fully accepted.

Schema documents are identified by the file extension xsd. Schema documents are enclosed by the root element Schema.

```
<Schema targetNamespace = "http://imert24.cims.rit.edu"
                xmlns:xsd = "http://www.w3.org/1999/XMLSchema">

</Schema>
```

The above is a Schema template used to hold schema declaration used in xml documents. The data types used in schemas have to conform to the vocabulary defined by the W3C consortium. In out example above the namespace xsd indicates that the document has to conform to the standard defined by the W3C Consortium. The attribute targetNamespace is the URL where the schema will be used.

Now we will go on discussing the features of schemas, which are:

Complex Type Definitions:

Complex type definition is the base contained to hold all the other elements in a schema. Complex Type can hold any other elements, text, or attribute. Complex Type is represented using the keyword ComplexType key in a schema definition.

```
<ComplexType name  = "Product">
        <element name="ProductName"  type="String">
                <attribute name=ProductID"  type="String">
         <element ref="datereceived"  minOccurs="0">
        <element  name="Units"  type="String">
        <element  name="Price"  type="String">
</ComplexType>
```

The above example illustrated that the instance of element Product declared will have three elements and an attribute, each of the type string. There is also an optional element, which references the element datereceived and holds an attribute value of zero. Complex type can contain other complex types as well.

Using Existing Definitions:

Schemas enable us to use types, which have been declared earlier in the document. If we refer to our complex type <Product> in the earlier example we have a reference to an element <datareceived>. So, this element <datereceived> can be declared at any other instance inside the document as:

```
<element name = "datereceived" type ="date" />
```

The attribute minOccurs=0 indicates that the element is optional in the schema definition.

Simple Types:

Elements which are declared directly and which do not contain any other elements or attributes are declared as Simple Types using Schemas. Simple Types can be considered to be basic building blocks of schemas. Simple Types are declared using the keyword SimpleType inside a schema. Schema Types can either be derived from other schema types or can use Primitive data types. Primitive data types are the standard data types as String, Boolean, float, double and many others as in procedural programming languages.

```
<SimpleType name ="configuration" type="string"/>
```

The above is an element <configuration> declared as a SimpleType in a schema definition.

Facets:

Facets are used to create new simple types modifying the base types (Primitive or derived data types).

```
<SimpleType name = "password"  base = "String"
        <minLength value = '4'
        <maxLength value = '10'
</SimpleType>
```

The above example uses facet where element <password> is derived of type string but has to four to six characters long to be valid.

The Pattern Facet:

Pattern Facets derives from base type string and is similar to regular expressions.

```
<SimpleType name = "passcode" base: "string" >
        <pattern value = "[a-z]" />
</SimpleType>
```

The above pattern declares an element passcode and signifies that only the lowercase letters of the alphabet can be used as passcode. Pattern facets can be used in creating passwords, zipcode, social security numbers and telephone numbers.

Enumeration facets:

Enumeration facets are used to defined enumerated data types inside schemas. Any Primitive data types except Booleans can be used in creating enumeration facets.

```
<SimpleType  name="tickerSymbol" base: "string">
        <Enumeration value = "KLM" />
        <Enumeration value = "MCC" />
        <Enumeration value = "DGD" />
</SimpleType>
```

The above is an element <tickerSymbol> that is declared as Enumeration facet and can only accept three possible values KLM, MCC and DGD.

Attributes:

Attributes are used to describe more details on elements.

```
<attribute name=ProductID"  type="String">
```

The above is an attribute ProductID that is used to describe the element Product.

The Content Attribute:

Only Complex Types in Schema declarations are eligible to hold attributes. The content attribute in Complex Type determines whether the complexType can contain only elements (elementOnly), cannot contain anything (empty), can content both text and elements (mixed) or can contain only text (textOnly).

<ComplexType name='productInfo' context='mixed'>

The above line declares an element <productInfo> as a complexType and can contain both text and elements (mixed).

Groups:

We have discussed earlier that Complex Types are can hold more than one elements. Groups in schema definitions are used to describe the order and occurrence of elements inside ComplexType declarations.

```
<ComplexType name='fullname'>
        <group order = 'sequence'>
                <element    name="firstname"   type="String"
                <element    name="lastname"    type ="String"
        </group>
</ComplexType>
```

The above declaration describes the sequence of a person's fullname. It indicates that firstname has to be followed by the last name. The sequence is described by the element <group>.

Derivation:

The biggest power of schema is the ability to derive new data types from existing data types, which is called inheritance in object-oriented terminology. New Elements can be derived either by extension or restriction. When deriving by extension new types are added to the base type and when deriving by restriction we override the existing types of the base.

```
<element name = "productweight">
        <ComplexType base = "integer" deriveBy="extension">
        <attribute name = "units" type = "integer" />
        </ComplexType>
</element>
```

The above is an example of inheritance (Derivation) in xsl Schemas. We have an element <sizeofproduct> that is derived (deriveBy) from the base class integer by extension. Now the element <sizeofproduct> contains attribute units, which is of the integer type.

4.3.3. XML Namespaces

XML is extensible in nature but extensibility does not come without constraint. When users have the flexibility of defining their own tags then there can be conflicts in-between the tags defined. The distribute nature of the web adds more conflict when defining different sets of tags.

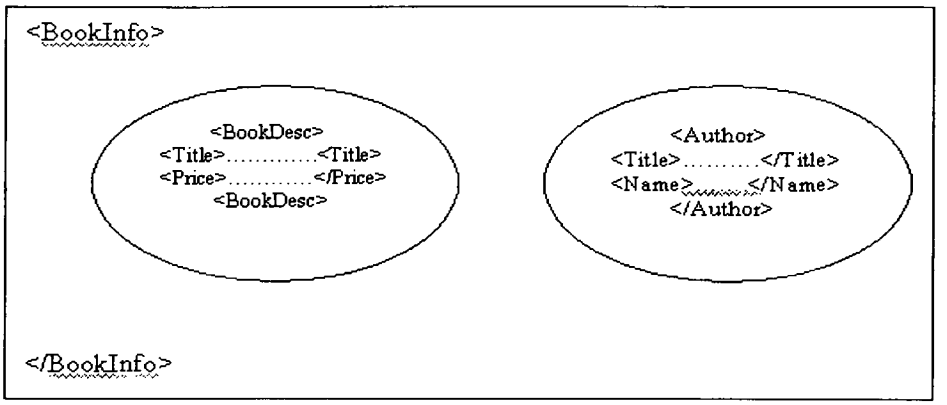

Figure 4.7: xml document, which stores Book Information

In the above figure 4.7, we have an xml document used to store book information. We can see in the figure that the tag <Title> is used twice in the same document. The tag <Title> is used once to describe Book Title and then used again to describe the title of the author. This creates a conflict with the same tag <Title> having two different meanings inside the same document.

Namespace is the mechanism to solve the problems of ambiguity, which can occur as in the scenario described above. Namespaces are the mechanism used in xml specification to identify elements uniquely.

```
<?xml version="1.0"?>

<Bookdsc:BOOKINFO xmlns="http://www.puskaradhikari.com">

        < Bookdsc:BOOK>
                <Bookdsc:TITLE>XML Developer's Guide</ Bookdsc:TITLE >
                <Bookdsc:PRICE currency="US Dollar">44.95</ Bookdsc:PRICE >
        </Bookdsc:BOOK >

        <AUTHOR>
                <TITLE>Mr.</TITLE>
                <NAME>Joe Smith</NAME>
        </AUTHOR>


</Bookdsc:BOOKINFO>
```

Figure 4.8: Using namespaces to describe book information

In Figure 4.8 namespace has been used to uniquely identify the different meanings associated with the element <TITLE>. The Bookdsc elements are identified using the namespace, <Bookdsc: BOOKINFO xmlns="http://www.puskaradhikari.com">. Since we are using only two primary elements to describe book, Bookdsc and AUTHOR, AUTHOR element will become unique by default.

Uri's (Unique Resource Identifier) are used to uniquely identify elements when using namespaces. URL's belonging to the author's domain name can be used as Uri's, which are very likely to be globally unique. For example, in Figure 4.8, the URL http://www.puskaradhikari.com has been used to uniquely identify the element Bookdsc.

4.3.4. Displaying XML

XML unlike HTML is designed to describe data and is a format for storing structured data. However, XML has no built-in method of displaying data to the users. However there two major technologies, which can be, used to display data to the user which are CSS(Cascading Style Sheet) and XSL (Extensible Stylesheet Language).

4.3.4.1. Cascading Style Sheets:

Cascading Style Sheets were originally designed for the use in Dynamic HTML documents. Cascading Style Sheets is a method to control the layout and design of our documents. Cascading Style Sheets, developed by Håkon Wium Lie and Bert Bos of the World Wide Web Consortium, offer a powerful and manageable way for authors, artists, and typographers to create much-requested visual effects that will put aesthetics to the forefront of the Web. (http://wdvl.com/Authoring/Style/Sheets/). Use of Cascading Style Sheets in documents facilitates Dynamic HTML.

Cascading Styles Sheets were designed to add style to HTML documents but can also be use in displaying xml documents to users. There are two versions of Cascading Style sheets available which are CSS1 and CSS2. Different browsers support cascading Style Sheets to different levels. Internet Explorer 5 and Netscape 6 have full support for all the different versions of Cascading Style Sheets.

There are different ways of including Cascading Style sheets inside a document to display data to the user and they are:

Inline Styling:

CSS can be used inline using the keyword STYLE in displaying an xml or HTML document to the browser.

```
<P STYLE="color: Red font-family: Times New Roman" >
This is an example created by a child of Nepal.
</P>
```

In the above declaration the phrase 'This is an example created by a child of Nepal' will be colored red and will be displayed using the Times New Roman family. Inline Style sheet is only defined for a particular instance of an element only. For example, in out example the style will be effective for the particular paragraph only.

Embedded Style Sheets:

A style may be embedded inside a document using the keyword STYLE.

```
<Style type="text/css">
<!--
 BODY  {background: color: white }
 P       {font: 8pt arial;}
-->
</style>
```

The above is a style sheet declaration, which can be embedded inside an xml or HTML document. The embedded style sheet declaration indicates that all the <P> tags will have be displayed using a font size of 8 and arial font family. The background color of the document will be red.

Embedded Style Sheets are useful for maintaining styles within a particular document. This provides a document level reuse of style.

External Style Sheets:

External Style Sheets are used in web applications where uniform styles have to be maintained. External styles Sheets are files with css extension, which contain style

information to be applied into html or xml documents. External Style Sheets are referenced into an xml or HTML document using the LINK keyword.

<LINK REL=Stylesheet HREF="mystyles.css" TYPE="text/css">

The above declaration is used in referencing an external Style Sheet file mystyle.css from an xml or HTML document. Using External Style Sheet provides more flexibility as various pages within a web application can share the styles.

Imported Style Sheets

This is a way of importing style sheets from other URL's. This can be accomplished by using the command @import.

```
<STYLE TYPE="text/css">
<!--
  @import url (http://www.msn.com/style.css);
  @import url (http://www.rit.edu/stylesheets/punk.css);
-->
</STYLE>
```

In the example above we are importing style sheets used by two URL's http://www.msn.com and http://www.rit.edu.

4.3.4.2. XSLT-Extensible Style Sheet Transformation

(XSL) Extensible Style Sheet is similar to CSS but processes data differently. XSL works differently because it transfers data into another structured format before displaying it to the user. This process of transferring data into structured format before displaying it to the user is called XSL Transformation (XSLT). XSLT are XML documents themselves that are used to transform other xml documents into user readable format. Extensible Style Sheets are written as templates, which conform to the set of vocabularies defined by W3C consortium. The latest set of XSLT vocabularies defined by the W3C Consortium is available at http://www.w3.org/TR/xsl/.

XSLT Style Sheet Example

The below is a complete stylesheet displaytable.xsl used in displaying the contents of xml file products.xml (Figure 4.4) into an html table (Figure 4.9).

```
<?xml version="1.0"?>
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">

  <HTML>
```

```
   <HEAD/>

<BODY bgcolor="ghostwhite">
<BR/>
<BR/>


<TABLE align="center" bgcolor="#0066FF" width="400">
        <tr>
          <td valign="top" bgcolor="#0066FF">
          <font color="white">
          <B>Product ID</B>
          </font>
          </td>

          <td valign="top" bgcolor="#0066FF">
          <font color="white">
          <B>Product Name</B>
          </font>
          </td>

          <td valign="top" bgcolor="#0066FF">
          <font color="white">
          <B>Units</B>
          </font>
          </td>

          <td valign="top" bgcolor="#0066FF">
          <font color="white">
          <B>Unit Price</B>
          </font>
          </td>
          </tr>

    <xsl:for-each select="Products/Product">
    <TR>
    <xsl:attribute name="STYLE">
    background-color: <xsl: eval>whichColor(this)</xsl:eval>
    </xsl:attribute>
     <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
      <xsl:value-of select="ProductID"/>
     </TD>

     <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
      <xsl:value-of select="ProductName"/>
     </TD>
```

```
          <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
            <xsl:value-of select="Units"/>
          </TD>

          <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
            $<xsl:value-of select="Price"/>
          </TD>
        </TR>
        </xsl:for-each>
    </TABLE>
    <BR/>

    </BODY>
    </HTML>
  </xsl:template>
  <xsl:script><![CDATA[
    function even(e) {
      return childNumber(e) % 2;
    }
    function whichColor(e) {
      if (even(e))
        return "#efefef";
      else
        return "#ffffff";
    }
  ]]></xsl:script>

</xsl:stylesheet>
```

Product ID	Product Name	Units	Unit Price
1000	Norton Bolts	2	$33
1001	EMG Conductors	5	$1500
1002	Sony Radio	2	$150
1003	Microsoft .Net	4	$249

Figure 4.9: Resulting HTML table after using XSL on products.xml

The latest specifications for XSLT by W3C Consortium enable the use of HTML tags inside the XSLT template. However, looking at the XSLT template displaytable.xsl we

have defined the
 tag as
. A tag like
 indicates that it is empty according to the xml specification. XSLT conforms to the xml standard so we have to close the tag to make it a well-formed xml document.

<Xsl: stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

The URL http://www.w3.org/TR/WD-xsl conforms that the stylesheet template used conforms to the vocabulary defined by the W3C Consortium.

<Xsl:template match="/">

The "/" in the above line tells the processor that this node applies to the root level of the document. The root level can be thought to be the imaginary pair of tags surrounding the entire of our xml documents (products.xml).

<xsl:for-each select="Products/Product">

The above line loops through each an element, which are inside the <Product> element. (<ProductID>, <ProductName>, <Units> and <Price>).

<xsl: value-of select="ProductName"/>

The above line embeds the selected item into the document as text. In this case it will embed the Product name, which is contained by the element <ProductName>.

```
<Xsl:script><![CDATA[
  function even(e) {
    return childNumber (e) % 2;
  }
  function whichColor(e) {
    if (even(e))
      return "#efefef";
    else
      return "#ffffff";
  }
]]></xsl:script>
```

Scripts can be used inside XSLT by placing them inside CDATA sections as above and called from inside the XSLT template. XSLT uses JavaScript by default. The Scripts above alternate the row color the resulting html table.

4.3.4.3. Comparing CSS and XSLT

Cascading Style Sheets are subsidiary of HTML but can still be used in displaying xml data to the browser. Plain HTML cannot be used to create well-formed documents.

However, the new generation of HTML, XHTML uses the existing HTML tags and conforms to the xml standard. Unlike CSS, XSLT itself is a well-formed xml document and cannot work well with unformed structure as HTML. However, CSS is designed to work with HTML.

CSS works better with XML but cannot be processed on the server side. CSS also does not also create any Markup structure. However, XSLT can be used both on the client and the server side. However, many browsers still do not have full support for XSLT on the client side. Companies like Microsoft have already incorporated XSLT into their latest release of XML parser.

CSS is designed strictly to work with browsers. However, XSLT is not designed for browsers only. XSLT can be used in displaying data into various formats such as text, PDF many others. XSLT can also be used to transfer xml into other xml format but with a different structure. XSLT is a flexible way of structuring data suited to different clients and enterprises. XSLT can be used to convert in-between the different vocabularies and standards so that enterprises can communicate upon each other.

4.3.5. The Parser

Every xml application we develop is based on a parser. Parser is a program which acts as an intermediately between an application and an xml file. Parser is a low level tool, which sits in-between our application and the xml file. The xml format is fairly simple yet there has to be a parser to validate the xml files. Parsers are similar to compilers, which validate syntax and grammars in a document.

There are two different kinds of xml parsers which can be classified as Validating and non-validating parses. Both of these kinds validate the correct use of syntax in the document but validating parsers validate against the document type declaration (DTD).

The parser reads the xml files and creates a tree in the memory, which matches the exact structure of the xml file. Then the application manipulates the tree structure in the memory. The application will interact with in-memory representation of the xml file instead of the xml file directly. The application will look at the in-memory representation as the entire document.

There are two ways to interface in-between the parser and the application, which are:

Object-based Interface:

Object-based Interface is the method the parser uses to build an in-memory representation of the tree in memory. The interface explicitly builds a tree in the memory before enabling the application program to manipulate it. The parser creates a duplicate tree structure of the xml file in the memory.

Event-Based Interface:

Event-Based Interface is another approach taken by the parser to interact with the application. The Event-Based Interface takes a more step-oriented approach. It does not build an explicit tree as the Object-based Interface. An event based Interface generates events as it finds specific symbols in the file.

The Object-based interface is good for the application as it can directly interact with the tree in the memory. However, Object-based interface involves more overhead as it keeps a duplicate copy of the xml file in the memory. The Event-based approach seems to involve less overhead but it makes things difficult for the application.

The above two different interfaces above Transfer into two different standards which are

- Document Object Model
- Simple API for XML

Different vendors have different parsers available to process xml. However, all of them confirm to the open satndards above the DOM or the SAX. The W3C consortortium has standard recommendations for both the DOM and the SAX standard.

4.3.5.1. Document Object Model:

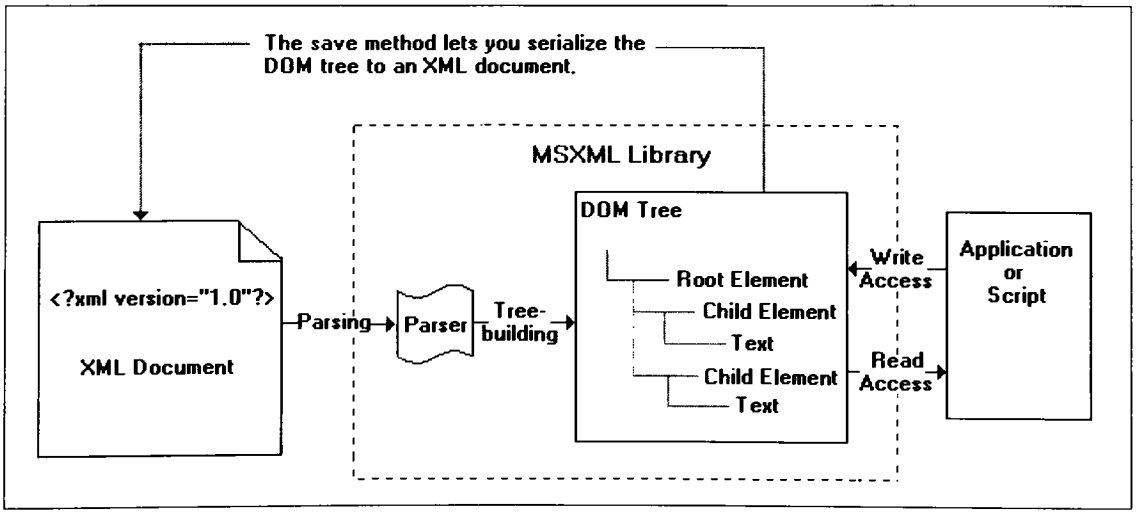

Figure: 4.10, illustrating Microsoft's DOM API (http://www.msn.com)

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents (http://www.w3.org/DOM/). This uses the Object-based Interface approach and creates an in-memory tree structure in the memory so that the elements can be accessed randomly from the application.

If we refer toFigure 4.10, the DOM tree is considered to be the document Object and the root element is called the documnetElement. The tree has nodes as element, attributes, text, child , parents and siblings. Differnet vendors provides their own sets of interfaces for manipulating the DOM object but comply with the same open standard Document Object Model.

4.3.5.2. SAX API:

The Simple API or SAX is based on the notion of the Event-based Interface. SAX requires less memory compared to DOM, as it does not build a tree in the memory. It is better to use SAX where we need to process small amount of document. So, SAX is a complementary DOM not an alterative. DOM is preferred in parsing small sized documents that do not require large space.

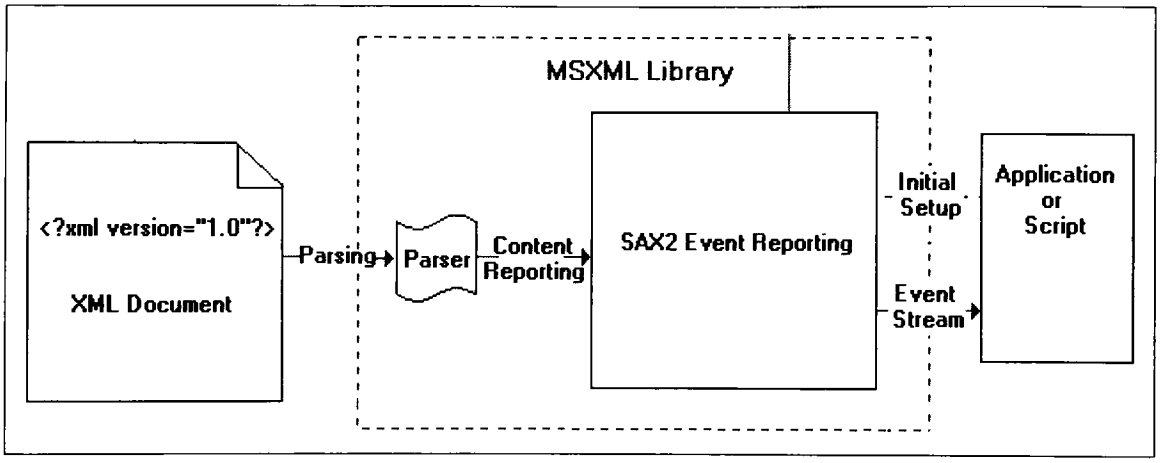

Figure: 4.11, illustrating Microsoft's SAX API (http://www.msn.com)

Figure 4.11, shows the architecture of the SAX processing. It does not build a memory tree but builds events as it comes across the symbol. SAX is better API to be used with large sized documents, which are not feasible to be loaded into the memory.

References

"Cascading Style Sheet". Web Developer's Virtual Library.
<http://wdvl.internet.com/Authoring/Style/Sheets/>.

"Edi For the Book World". EDTtEUR. < **http://www.bic.org.uk/infopak.html**>.

"Frequently asked question about extensible Markup Language". The XML FAQ.
<**http://www.ucc.ie/xml/**>.

"Level 1 Document Object Model Specification" W3C. < **http://www.w3.org/TR/WD-DOM/**> July 20, 1998.

"Welcome to XML School" XML Tutorial. <**http://www.w3schools.com/xml/default.asp**>.

Webopaedia; Computer Related Internet Dictionary. < www.webopedia.internet.com>.
Word Wide Web Consortium. <www.w3.org/xml>.

"Extensible Markup Language (XML) 1.0 (Second Edition)" W3C. < http://www.w3.org/TR/REC-xml> (October 6, 2000).

"XSLT, XPath and XSL Formatting Objects". Web Developer's Virtual Library.
<**http://wdvl.internet.com/Authoring/Languages/XSL/**>.

"XML/EDI the e-business framework" XML/EDI Group Homepage. <**http://www.xmledi.com**>.

Bosak, Jon. "XML, Java and the Future of the Web".
<http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm>.

Bos, Bert. "XML in Ten Points" <www.w3.org/xml/1999/xml-in-10-points> (March 09 2000).

Buczek, Greg. Instant ASP Components (Application Development). New York: McGraw Hill, 2000.

Bradley, Neil. The XML Companion. New York: Addison Wesley, 2000.

Cartwright, Adam S. "Server-Side XML in ASP" < http://www.15seconds.com/issue/990527.htm>

Matt J, Crouch. Web Programming with Asp and Com. New York: Addison Wesley, 1999.

Ducharme, Bob. "HTML and XSLT" <http://www.xml.com/pub/a/2000/08/30/xsltandhtml/index.html >(August 30, 2000).

Ducharme, Bob. "Namespaces and XSLT Stylesheets"
<http://www.xml.com/pub/a/2001/04/04/trxml/index.html> (April 04, 2001).

Duncharme, Bob. "Adding Elements and Attributes".
<http://www.xml.com/pub/a/2000/08/02/xslt/index.html> (April 02, 2000).

GinsBourg, Niall. "Creating an In-Memory Database Using XML and XPath -- Part 1"
< http://www.15seconds.com/Issue/010409.htm>.

Goldfarb, Charles, and Prescod, Paul. The XML Handbook, 2nd Edition. New Jersey: Prentice Hall, 2000.

Goldfarb, Charles; The SGML Handbook; Oxford: Clarendon Press, 1990.

Holman, Ken G. "What is XSLT" <http://www.xml.com/pub/a/2000/08/holman/index.html>.

Howe, Walt. 'A Brief History of the Internet'. < http://www0.delphi.com/navnet/history.html>.

Homer, Alex, et al. Professional Active Server Pages 3.0. Birminghan: WROX Press, 1999.

Claben, Michel. "XML - the better HTML?" <www.webreference.com/xml/column1/index.html>.

Kotok, Alan. "XML to the rescue". <http://www.xml.com/pub/a/1999/08/edi/index3.html> (August 04, 1999).

Kropog, Bill. Professional ASP XML. Birminghan: WROX Press, 2000.

Kyrnin, Jenifer. "HTML is dead? Long Live HTML!"
<http://html.about.com/compute/html/library/weekly/aa031501a.htm>

Lam, John. "Visual Interdev and ASP".
<http://www.zdnet.com/pcmag/pctech/content/16/22/pp1622.001.html>

Libre, Juan. Beginning Active Server Pages 3.0. Birminghan: WROX Press, 1999.

Megginson, David. "XML-The Big Deal?" < http://www.futureofsoftware.net/dm2_0010/dm0010.asp>.

Miller, Ken. Inside Microsoft Visual Interdev. Washington: Microsoft Press, 1998.

Mohr, Stephen. Designing Distributed Applications With XML, ASP, IE5, LDAP and MSMQ. Birminghan: WROX Press, 1999.

Munro, Alistair. 'Moving To Distributed Processing Standards ... Remote Procedure Call'
<http://www.ja.net/documents/NetworkNews/Issue44/RPC.html>.

Price, Andrew. "Creating Dynamic Web pages with XML/ASP & XML format".
<http://www.vbxml.com/xsl/articles/dynamic/dynamic.asp>.

Sperberg-McQueen, C.M. "XML-Related Activities at the W3C".
<http://www.xml.com/pub/a/2001/01/03/w3c.html>. (January 03, 2001).

Vander Vi List, Eric. "Using W3C XML Schema".
<http://www.xml.com/pub/a/2000/11/29/schemas/part1.html> (November 29, 2000)

Vink, Ian. "Distributing Server Load to the Client with XML and XSL".
<http://www.15seconds.com/issue/010305.htm>.

Walsh, Norman. "A Technical Introduction to XML".
<http://www.xml.com/pub/a/98/10/guide0.html>.

5. Simple Object Access Protocol

5.1. Introduction to SOAP-Simple Object Access Protocol

SOAP is a protocol specification for invoking methods on servers, services, components and objects (http://xml.coverpages.org/soap.html). Internet has grown exponentially over the past decade and has been an integral part of everyone's life. The WWW was designed with a limited set of HTML tags to display data contents over the browser. The World Wide Web is an Information System, which relies primarily upon the HTTP Protocol and has ties itself with many other common Internet protocols (FTP, SMTP, NNTP etc). But, the WWW has evolved from its original idea of displaying static contents from a web server to the browser. Today the WWW has evolved into Information systems, which needs to work interactively with various database systems, heterogeneous components, operating systems and platforms. Integrating and sharing information between various systems and platform has been a prime effort over the past few years. There have been technologies like CORBA, DCOM and Java RMI, which seem to contribute to the information-sharing module through the implementation of distributed object Architecture. However, these technologies have appeared to be a failure when it comes to integrating various platforms, languages, existing legacy systems and passing across corporate firewall boundaries. The highly distributive nature of the Internet requires the integration of various platforms, systems and languages for successful information sharing.

SOAP-Simple Object Access Protocol is designed to fulfill the demands of the distributed Architecture of the World Wide Web. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment (http://www.w3.org/TR/SOAP/). SOAP is a Protocol that incorporates data into XML payload and transport it using the HTTP protocol over the World Wide Web. SOAP, was designed to be used with the HTTP Protocol over the WWW. However, the latest specification of SOAP gives the flexibility of transporting SOAP message using other standard Internet protocols as FTP.

5.2. History of SOAP

Dave Winer from Userland Corporation originally published SOAP in the April of 1999 to use xml for establishing RPC (Remote Procedure Call) over the Internet. The first standard specification for SOAP v0.9 was introduced in September of 1999. This draft standard, which specified the use of XML and HTTP as an RPC-like data communication infrastructure, was intended to provide a mechanism for distributing objects over the Internet (Scribner and Stiver, 2000). The first standard draft for SOAP v0.9 was not able to fulfill the demands of the developers so v1.0 was released to alleviate the problem. The latest specification for SOAP, SOAP 1.1 (http://www.w3.org/TR/SOAP/) inherits from existing XML-RPC protocol. Actually existing XML-RPC protocol was previously known as SOAP before being published by the W3C Consortium as XML-RPC. SOAP is a superset of XML-RPC, but they are not compatible. (http://soap.weblogs.com/faq). SOAP is a young Protocol and the specifications are still work under progress. SOAP has not yet been used by major enterprises as XML-RPC Protocol. Microsoft took a

major role towards the publicity of SOAP but major vendors like SUN, IBM have slowly started putting their hands onto SOAP.

XML-RPC and previous specifications of SOAP as SOAP 0.9 and SOAP 1.1 supported communications only between two endpoints. However, the latest specification for SOAP provides the flexibility of transmitting SOAP message between various endpoints before reaching the final destination.

5.3. The Foundations of SOAP (XML and HTTP)

SOAP is a XML based protocol, which relies upon HTTP for transporting messages in-between different systems in the Internet. SOAP is primarily designed to execute Remote Procedure Call. The latest specification for SOAP v1.1 enables SOAP to be used with other Internet protocols as SMTP and FTP. (XML) Extensible Markup Language and HTTP (Hypertext Transport Protocol can be considered to be the basic foundation of SOAP.

In the past different formats have been used to exchange electronic information. The major techniques used over the past years includes:

- Proprietary Formats
- Spatial and Referential Schemas and
- Structured Data Scheme

Proprietary Format:

This techniques assumes that both the endpoints involved communication agree with the same format. The serialization and de-serialization process with Proprietary format enforces a particular order or sequences. The exact sequence has to be flowed at both ends. This kind of format is not open and not very flexible. This technique leaves no room for errors. This is where communication protocol like SOAP comes into play. One of the major goals of SOAP is to standardize an open format, which is not vendor and format specific as Proprietary Format.

Spatial and referential Schemas:

This is a messaging format, which is used by conventional networking protocols as TCP/IP. The spatial and referential format requires the data to be organized into particular messaging format. This format keeps the data structure open and flexible.
For example, Figure 5.1 describes of the structure of TCP/IP Packet structure, which uses the spatial and referential approach in building and sending message packages. The structure of the packet is described organized into layers. Since, HTML focused only on displaying data to the client, a standard was required which would give meaning to data. Extensible Markup Language (XML) was created as a subset of SGML to give more meaning to data as opposed to HTML.

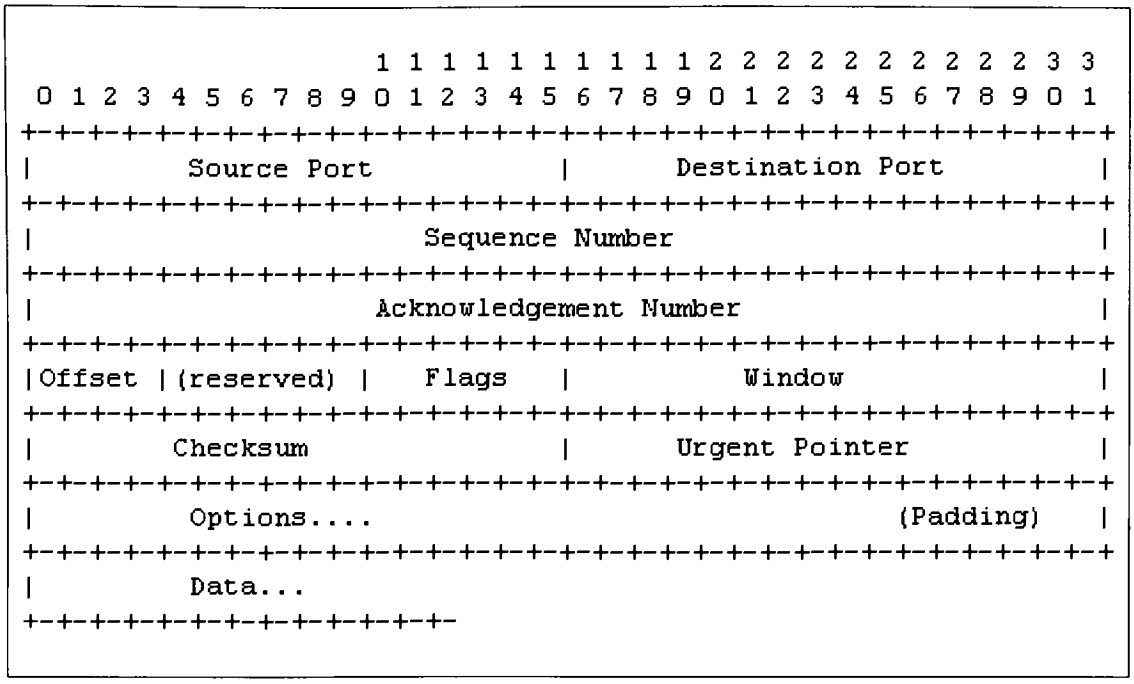

Figure 5.1: TCP/IP Segment (http://www.garykessler.net/library/tcpip.html)

Given a particular stream of data, identifiers are used to specify that particular information immediately follows or references data in another location (Scribner and Stiver, 2000). These kinds of system are open, provide some degree of structure, and involve fewer overheads. However, this approach faces problems such as size limitation and is not designed to be extensible.

This is one of the reasons for the foundation of such a protocol as SOAP. SOAP was founded upon extensible data streams that are easy to read and write, and formats that can change at will without negatively impacting applications (Scribner and Stiver, 2000).

Structured Data Scheme

With the growing use of Internet over the past decade organizing data into structured format for communication has proven to be a de facto standard for communication. Markup Languages came into use in order to organize data into structures. World Wide Web was invented with the purpose of displaying statically stored data to the browser (client) from a web Server. The WWW structured data into HTML (Hypertext Markup Language) and transported it using the HTTP (Hypertext Transport Protocol) to the client (Web Browser). XML was designed with the goal of simplifying data communication over the Internet, support wide variety of applications, providing inheritance features to data and documents (Extensibility) and give meaningful approach to structured data. So, SOAP can be considered to be a protocol derived from XML, which is intended to promote distributed, platform independent and language neutral Internet communication.

SOAP was founded with the purpose of serializing and de-serializing data into using XML packet and transporting it within and across network boundaries using HTTP. The goal of SOAP is to enable RPC communication between heterogeneous systems using intelligible languages and protocols as HTTP and XML. The intention of SOAP is to integrate "incompatible" systems minimizing deployment and development time. SOAP creates two layers within each message packet Information Layer (XML) and a Transport layer (HTTP) in achieving Internet based communication.

5.4. Disadvantage and Advantages of SOAP

Like anything else SOAP also has advantages and disadvantages. SOAP is a growing protocol that faces many drawbacks at the current moment. However, the growing involvement of Organizations and developers in the SOAP community makes us believe that the drawbacks will be resolved within the next coming years. The major advantages of a protocol like SOAP are:

- SOAP is a lightweight protocol. Other technologies like DCOM, CORBA and Java RMI make use of Heavyweight protocols that require their own runtime environment to be present at both client and the servers to be functional. However, SOAP is built upon fundamental capabilities and universally available protocols as XML and HTTP, which makes it a lightweight Protocol. There is no heavy-duty development and Deployment effort associated in implementing SOAP.

- SOAP is not vendor specific. However, other distributed object technologies as DCOM and CORBA are vendor specific. No single vendor dominates the SOAP specification. It is built up on open standards as HTTP and XML.

- Minor changes to SOAP infrastructure will not affect the applications that are already using SOAP protocol in their implementation. No major changes have to be made even if the version of SOAP changes.

- With the new specification for SOAP, v1.1, it opens room for existing Internet protocols as IIOP and RMI to be used in communication. The previous specifications used only HTTP in transporting SOAP contents.

- SOAP is platform independent and language neutral solution to Remote method Invocation. SOAP is designed upon a notion that any operating system as well as any third generation languages can interpret HTTP and XML.

SOAP is still in its preliminary phase and is still under developmental stage. It is a new protocol, which is gaining slow acceptance. The drawbacks SOAP is facing currently are

- SOAP is a new protocol and the interoperability of SOAP is not totally feasible. SOAP has not yet been tested in diverse enterprise level applications. However,

the good new is that W3C is setting up a team with the aim of standardizing xml-based protocol as SOAP. In addition, the OMG has set itself the task of making CORBA compatible with the communication protocol (http://idm.interment.com).

- SOAP uses XML schemas in its serialization process. However, XML schemas are still a work on progress and changes frequently. There are also no truly validating XML processors available with full schema support at this moment. However, this does not mean that SOAP will become. Actually the future of SOAP depends upon the involvement of the Developer communities and organizations.

- The current specification of ship does not support serialization by reference. Although serialization by value isn't inherently bad, it does mean multiple copies of the object will, over time, contain state information that is not synchronized with other dislocated copies of the same object (Scribner and Stiver, 2000). It is strongly believed that existing distributed Object architectures as DCOM and CORBA can be easily incorporated with SOAP. The overhead and space involved will be reduced if serialization can be achieved by reference.

5.5. Structure of the SOAP Message

SOAP is a lightweight protocol, which encapsulates xml message using the HTTP protocol and transport in the network. SOAP messaging structure can be basically classified into two parts, which are:

- Hypertext Transport Protocol (carrier) and
- Extensible Markup Language (content).

5.5.1. HyperText Transport Protocol (HTTP):

The HyperText Transport protocol is the de facto standard for transferring document for the World Wide Web. The latest specification for the HTTP protocol, Version 1.1 enables the transfer of any kind of document using HTTP protocol. HTTP uses the TCP/IP protocol and operates on port 80 by default. Figure 5.2 below illustrates the basic structure of the HTTP protocols, which contains the element Post, host, content-type and content-length.

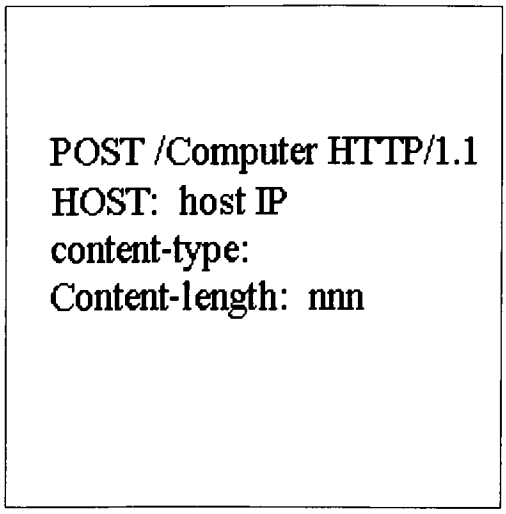

Figure 5.2: the basic structure of the HTTP protocol

There are two kinds of HTTP messages, which are:

- HTTP Request
- HTTP Response

5.5.1.1. HTTP Request/Response Model:

HTTP Request:

There are three kinds of HTTP requests, which are supported by the HTTP protocol. The three forms of http request are:

- **GET**: This is the most widely used method of HTTP Request. This method identifies provides the server with necessary information to identify a particular resource, so it can locate the data and send it to the client (Scribner and Stiver, 2000). The GET method transports the message content by using the client's URL in fulfilling the request/response model. A successfully invoked response to a request using HTTP GET method might look like:

  HTTP:/ 1.1 200 OK
  Content-type: text/html
  <HTML>
  <HEAD>Puskar's Website</HEAD>
  <BODY>HTTP Test</BODY>
  </HTML>

- **HEAD:** With the HEAD method of HTTP request/response model the client is able to receive only the HTTP header information of the server as a response.

- **POST**: This is another method used by HTTP protocol in sending a request message. The post method is primarily used in posting news and forms. The POST method encloses the message contents into the body of HTTP response/request packet during client/server communication.

HTTP Response:

This is the reply sent by the server to the client after receiving the request packet. A typical HTTP Response packet might look like:

```
HTTP/1.0 200 OK
Date: March 21, 2001
Server: Apache/1.3.0 (Unix)
Last- modified: January 01, 2001
Content-Length: 829291
Content-Type: text/html

<HTML>
<HEAD>Puskar's Website</HEAD>
<BODY>HTTP Test</BODY>
</HTML>
```

If we look at the first line of the response packet we see the line, HTTP/1.0 200 OK where 200 indicates the response status codes. The other possible response status codes are:

- **200 OK**, which indicates that the request was successful.
- **301 Moved Permanently**, which indicates that the object moved to a new location.
- **400 Bad Request**, which indicates that the server could not interpret the message.
- **505 HTTP version NOT Supported**, which indicates that the server does not support the http version.

5.5.1.2. SOAP Request/Response Model

According to the Internet Client/Server Model a client makes request to the server, the server fulfills the request and sends a reply back to the client. After the server provides the request it discards the state information and closes the connection and waits for another request. This client/server model is designed to involve fewer overheads and save resources. However, the latest version of HTTP, version 1.1 is designed to work in a different way. The server does not close the connection immediately after it fulfills the request as re-establishing connection frequently and for the same client becomes very expensive process. To handle this issue the server keeps the connection but discards all the state information after each and every HTTP transaction.

The SOAP inherits the architecture of the HTTP protocol in its request/response model. SOAP uses the HTTP POST method in transferring data between endpoints. This method is preferred because it carries HTTP request content in the body of the request and not on the URL as with the GET method. When a SOAP request is successful the server will send a reply back to the client indicating the requested information. However, if the server is unable to fulfill the request then it will indicate a failure message wrapped inside a XML payload which in SOAP vocabulary is referred to as FAULT.

SOAP and the Post Method:
SOAP makes use of the HTTP post method in sending a request to the server. The header in a SOAP request message might look like:

```
POST /myserver.asp HTTP/1.1
HOST: www.rit.edu
Accept: text/*
Content-type: text/xml
Content-length: nnn
SOAPAction: executeQuery#product
```

Looking at the SOAP Request above we notice that it inherits from the HTTP protocol. SOAPAction filed has been added in the header of the SOAP message. This field SOAPAction is mandatory and reveals the intent of the SOAP payload being sent to the server. This has been added to the SOAP message for security purpose.

SOAP and the M-POST Method:

In the previous version of SOAP specification v1.0, it was mandatory for clients to use POST method to make a SOAP Request call. If the request failed with an error code of *510 Not Extended,* then the client was eligible to use the M-POST method. However, the current version of SOAP specification, v1.1, enables the use of either POST or the M-POST method in making a SOAP request call.

The M-POST method is also an extension of the HTTP protocol and requires the presence of the MAN element in SOAP message header.

```
M-POST /myServer.asp HTTP/1.1
Man: "http://www.microsoft.com/protocols/text/SOAP"; ns=01
 01-MethodName: runQuery
Content-Type: text/xml-SOAP
Content-Length: 1234
```

The above is an example of a SOAP message header using the M-POST request method. The Man element consists of an URL address and a two-digit extension prefix (ns=01). The ns tag is similar to namespaces in xml documents. This is implemented for security

reasons and to identify messages that are eligible to pass across corporate firewall boundaries.

5.5.2. SOAP-XML Payload

SOAP is a Remote Procedure Call (RPC) mechanism, which involves XML. In a SOAP message the method parameter and data are encoded using XML to form a payload. The XML Payload in SOAP message servers the same purpose as the NDR (Network Data Representation) in DCOM and CDR (Common Data Representation) in CORBA. With SOAP the message is serialized and de-serialized at both communication endpoints using XML compatible software. To make SOAP work there has to be code running at both endpoints (client and server). Code has to be written at both communication end points to build and read SOAP message packets. The SOAP specification has been kept fairly simple so implementation should not be a major development effort and very expensive. SOAP packages can be created and interpreted by any kind of web and Object-oriented programming languages.

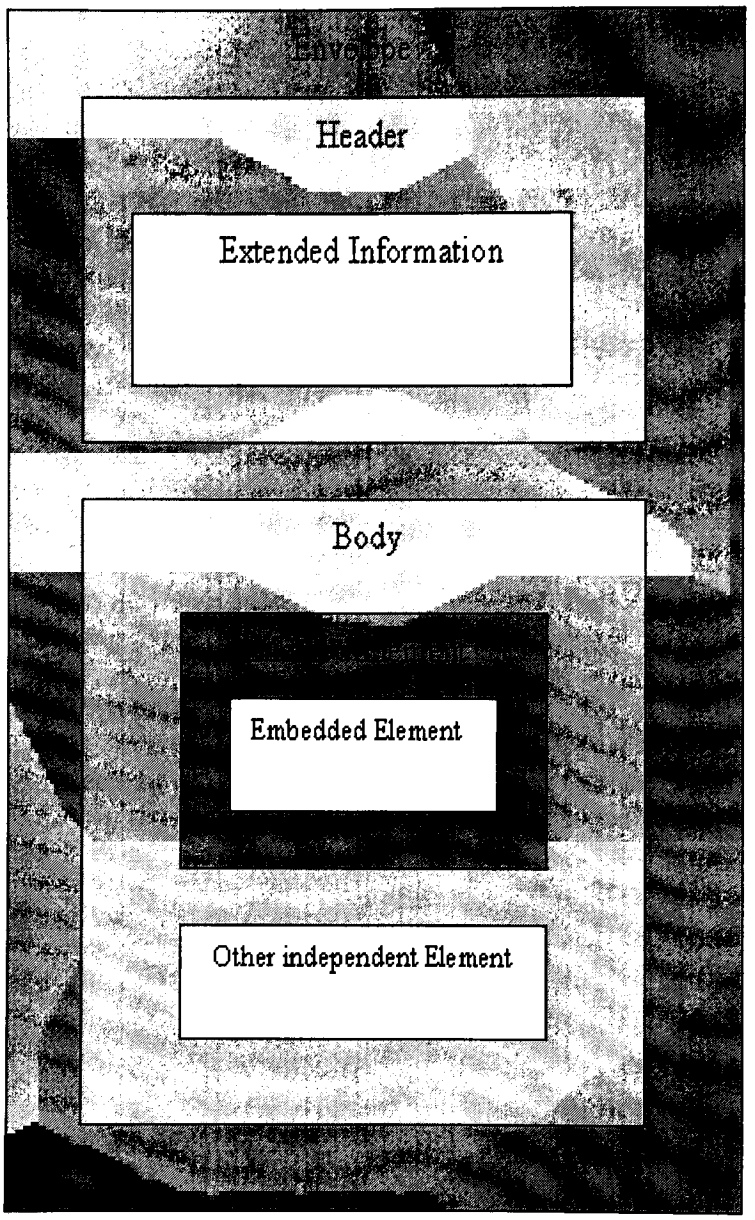

Figure 5.3, The Basic SOAP payload Structure (Scribner and Stiver, 2000).

The SOAP payload structure consists of three portions as:

- SOAP Envelope
- SOAP Header and
- SOAP Body

Figure 5.3, shows a pictorial representation of the basic SOAP payload structure. The SOAP Envelope wraps the child elements Header and Body inside it. The SOAP parsers residing at all the communication endpoints must be able to interpret and process the namespaces defined in the Envelope element. If the namespace is incorrect the message should not be processed and response packet built indicating the error. The namespaces used in the SOAP specifications are:

5.5.2.1. SOAP Envelope: The Envelope section of a SOAP payload structure uses the namespace identifier http://schemas.xmlsoap.org/soap/envelope/ to elements and attributes.

SOAP Serializer: The Serialization Element in the SOAP Payload specification uses the namespace identifier http://schemas.xmlsoap.org/soap/encoding/

The SOAP Envelope Structure:

If we look at the structure of any xml document we can see that the processing expression <? xml version="1.0" is required at the top. However, SOAP Payload does not require the presence of the xml processing instruction as illustrated in Figure 5.4 below. The SOAP payload considers the Envelope Element as its root element.

```
<SOAP-ENV: Envelope>
        <SOAP-ENV: Header>
        <! This Header element is optional in SOAP body
</SOAP-ENV: Header>


<SOAP-ENV: Body>
        <! This will contain the method name and parameters
</SOAP-ENV: Body>


</SOAP-ENV: Envelope>
```

Figure 5.4: illustrating the SOAP-Env data layout

The SOAP Envelope is free to contain additional namespace declarations. The SOAP element can contain additional sub-elements but they have to be qualified by namespace and follow the body element. The SOAP element must be present in a SOAP message. The elements and attributes in a

SOAP Encoding Attribute:

The SOAP Encoding Attribute is designed to indicate the serialization rules in a SOAP message. This attribute can be used with any attribute or element in a SOAP message. The URL http://schemas.xmlsoap.org/soap/encoding defines the standard namespace used for the SOAP Encoding Attribute. There is no default Encoding Attribute available for a SOAP message. SOAP makes use of XML schemas in its serialization rules.

SOAP Envelope Versioning model:

Any SOAP message, which does not have an Envelope element, associated with the namespace http://schemas.xmlsoap.org/soap/envelope/ is treated as an error. The Envelope Element is a mandatory part of a SOAP message.

5.5.2.2. SOAP Header

This is an optional mechanism provided by SOAP Specification to extend message in a decentralized way without the prior knowledge of both communication endpoints. The header element is an optional section of the SOAP message. The Header element has to be the first immediate child of the Envelope Element. It is essential for the header element to adhere to the SOAP specification rules unless the header has been modified using the SOAP-ENV: encodingStyle attribute.

The SOAP mustUnderstnad attribute can be used to indicate whether the header entry is an optional or mandatory for the recipient to process the message. A mustUnderstnad value of "1" indicates the processor that the header entry has to be present. A mustUnderstnad value of "0" indicates the processor that the header element is optional.

The latest version of SOAP specification enables SOAP messages to pass through various intermediaries before reaching the final destination. All the SOAP endpoints and the intermediaries have to be identified by a Uniform Resource Identifier. All the SOAP intermediaries and endpoints involved are identified using the SOAP Actor global Attribute.

5.5.2.3. SOAP Body

The SOAP Body element is a mandatory feature of the SOAP message format. The Body element of SOAP is usually used to marshal RPC calls and report errors. According to the latest version of the SOAP specification, v1.1, it indicates three types of body elements:

- **Request**: This is the packet, which contains essential information to conduct Remote Method Invocation call.

- **Response**: This is the packet, which will contain the information returned by the remote method after being invoked.

- **Fault**: The fault element is included incase of failure to invoke the Remote Procedure call.

The SOAP Request Call Body Element

If the SOAP message packet is a Request Call then the first child of the Body element contains the method name. Then the method arguments will be embedded inside the element containing the method name.

Float calculateAmount (numYears, interest, totalCost)

The above is an example of a C function calculateAmount, which takes in parameters numYears, interest and totalCost as arguments. If we are to call the C function above from a remote machine using SOAP the message packet containing serialized data in XML format will look like:

```
<SOAP-ENV: Envelope
        xmlns: myMethod="http://lathe.cims.rit.edu"
        xmlns: SOAP-ENV=http://schema.xmlsoap.org/soap/envelope
        SOAP-ENV: encodingStyle=http://schema.xmlsoap.org/soap/encoding>
        <SOAP-ENV: Body>
                <myMethod: calculateAmountRequest>
                        <numYears>1.5</numYears>
                        <interest>12</interest>
                        <myMethod: totalCost>1234.12</totalCost>
                </myMethod: calculateAmountRequest>
        </SOAP-ENV: Body>
</SOAP-ENV: Envelope>
```

The above SOAP request packet is equivalent to making a function call inside a C module as:

```
CalculateAmount (1.5, 12, 1234.12);
```

The method name is identified using the namespace identifier http://lathe.cims.rit.edu. However, the method arguments do not require the name space qualification.

SOAP Response Body Element:

The serialized SOAP message containing the serialized method arguments can be sent to the remote machine using its IP address. The remote machine will parse the SOAP message using its XML processor and invoke the method calculateAmount. The value returned by the function calculateAmount will be serialized into SOAP Response packet and sent back to the client. If there is no error and the method is invoked successfully the serialized SOAP packet will look like:

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 122

<SOAP-ENV: Envelope
xmlns: SOAP-ENV=http://schema.xmlsoap.org/soap/envelope
SOAP-ENV: encodingStyle=http://schema.xmlsoap.org/soap/encoding>
<SOAP-ENV: Body xmlns: myMethod="http://lathe.cims.rit.edu">
        <myMethod: calculateAmountResponse>
                <Return> 1286.12</Return>
        </myMethod: calculateAmountResponse>
</SOAP-ENV: Body>
</SOAP-ENV: Envelope>
```

The rules for serializing a Response are similar to the response call. The method name has to be identified using a namespace. However, the Return value does not require a namespace identifier. The SOAP Response packet above indicates that the method calculateAmount has been invoked successfully and returned a value of 1286.12.

The SOAP FAULT Body Element:

The role of the SOAP Fault element is to carry the information back to the client incase of any network or application. In this case the header of the SOAP message will indicate the internal server error and the error number. The SOAP fault Element consists of four fault sub-elements, which are:

- **Faultcode:** This Faultcode element is mandatory in a SOAP message indicating fault. An example of a Faultcode would be SOAP-ENV: Inter Server Error 500

- **FaultString:** This element FaultString is present to provide a human readable version of the SOAP error and is a mandatory part of a SOAP Fault message.

- **Faultfactor**: This is another mandatory element of the SOAP fault message. It is used to identify the computer, which caused the SOAP error to occur.

- **Detail**: The detail error is used in carrying application specific information related to the SOAP body element.

If the calculateAmount function encountered an application level error the SOAP Response will indicate fault, which will look like:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml
Content-Length: 122

<SOAP-ENV: Envelope
xmlns: SOAP-ENV=http://schema.xmlsoap.org/soap/envelope
xmlns: SOAP-ENV: encodingStyle=http://schema.xmlsoap.org/soap/encoding>
<SOAP-ENV: Body>
        <SOAP-ENV: Fault>
                <Faultcode>500</Faultcode>
                <FaultString>
                        HTTP version mismatch
                </FaultSring>
        </SOAP-ENV: Fault>
</SOAP-ENV: Body>
</SOAP-ENV: Envelope>
```

The above SOAP fault message is returning a HTTP version mismatch error. The SOAP header indicates that there is an internal server error. The details of the error are embedded inside Faultcode and FaultString elements.

5.6. SOAP and the Data Types

SOAP makes use of XML format in constructing and defining data types for encoding. SOAP specification puts constraints on how the data is to be serialized. With SOAP the data are divided into two groups, which are Basic SOAP data Type and Complex data Type. The data can be either single references accessor or multi-reference accessor. With single references assessors the data or arguments are accessed only once. However, with multi assessor the arguments are accessed multiple times.

Basic SOAP Data Types

The Basic SOAP data types can be categorized into various groups, and they can involve both single and multiples accessor. The basic SOAP data types are:

SOAP Simple Data Types:

SOAP Simple Data Types constitute of both primitive data types and XML defined data types. The primitive data types resemble data types, which are used in other procedural programming languages. The others are data types which are intrinsic to XML but which are used by SOAP with additional constraints. Figure 5.5, below is a summary of basic data types used in SOAP encoding.

Data Type	Meaning
String	XML character Strings(This is constrained by SOAP)
Boolean	True or false
Binary	Binary data
TimeInstant	String with data time combination
TimeDuration	Data showing year, day, month and time Combination
RecurringInstant	A particular Instant of time
Uri	Universal Resource Locator
Language	Language formats
Name	XML standard names
NMTOKEN	XML defined name toke type attribute
ID	XML ID attribute type
IDREF	XML IDREF attribute Type
ENTITIES	XML ENTITY attribute type
NOTATION	XML Notation attribute type
Decimal	Numeric values expressed in exact fraction
Integer	Numeric values without Fraction
Non-negative Integer	Inters values zero and greater
Positive Integer	Integer value greater than one
Non-positive Integer	Negative Integers including zero
Negative Integer	Integer value from one to negative infinity
Date	Time duration which lasts 24 hours starting at midnight
Time	Regular Time

Figure 5.5: List of SOAP simple data types.

SOAP Byte Arrays:

The latest version of the SOAP specification allows arrays to be encoded using opaque Base64 elements. The advantage of using these Base64 elements for serialization is that it looks at characters as the same whether they are ASCII or EBCDIC.

The below is an example of a string which has been encoded using the base64 encoding algorithm. This below example has been extracted from the current version of the SOAP specification available at http://www.w3.org/TR/SOAP/provides.

```
<picture xsi: type="SOAP-ENC: base64>
    aG93IG5vDyBicm73biBjb3cNCg==
</picture xsi: type="SOAP-ENC: base64>
```

Using the Base64 decoding algorithm the byte array in the above example will be read as *how now brown cow<CR><LF>*. The benefit of the Base64 encoding is there are no <CR>, <LF>,'.', or '-' characters that firewalls or other software protocols might find significant and therefore modify or otherwise molest (Scribner and Stiver, 2000).

SOAP STRINGS:

Strings are simple data types, which are designed to be multi-reference according to the SOAP specification. In this example we have a C function declared as

```
Char* string1 = "My Example1";
Char* string2 = "My Example2";

int  CompareString(char *string1, char *string2)
```

Expressing the above function in SOAP encoding format will look like:

```
<SOAP-ENV: Body xmlns:m='http://lathe.cims.rit.edu'>

        <m:CompareString>
                <string1 href="#str1"/>
                <string2 href="#str2"/>
        </m: CompareString>

<m:String1    id="#str1"     type="SOAP-ENC:string">
        My Example1
</m:String1>

<m:String2    id="#str2"     type="SOAP-ENC:string">
        My Example2
</mString2>

</SOAP-ENV: Body>
```

Enumerated Data Types:

Enumeration is defined to be a set of distinct names grouped together. The below is example of enumeration ***daysofweeks*** which binds the seven days in a week.

```
<Element name = "daysofweek" />

<SimpleType Name="daysofweek" base="xsd: string">

        <Enumeration value="Sunday"/>
        <Enumeration value="Monday"/>
        <Enumeration value="tuesday"/>
```

```
<Enumeration value="wednesday"/>
<Enumeration value="Thursday"/>
<Enumeration value="Friday"/>
<Enumeration value="Saturday"/>

</SimpleType>
```

SOAP Compound Data types:

SOAP specification extends the concept of simple data types by allowing us to encode compounding data types using the existing Simple data Types. The two different kinds of compound data types according to SOAP specification are Structure and Arrays.

Structures:

The current version of SOAP specification (http://www.w3.org/TR/SOAP/) defines a structure as:

"A 'struct' is a compound value in which accessor is name is the only distinction among member values, and no accessor has the same name as any other."

Structures are serialized multi-reference accessor. In the example below we have a C strut person declared as:

```
Struct person
        {
          String name;
          double height;
          double weight;
          int yearofbirth;
        } description;
```

The C struct person will be used in the main program as

```
description     info;
info.name = "Puskar Adhikari";
info.height=162;
info.weight=148;
info.yearofbirth=1976;
```

The above C strut serialized into SOAP encoding format will look like

```
<SOAP-ENV:BODY    xmlns:m="http://lathe.cims.rit.edu"?>
```

```
<m:info>
        <name xsi:type="string"> Puskar Adhikari</name>
        <height xsi:type=integer">162</height>
        <weight xsi:type="integer">148</weight>
        <yearofbirth xsi:type="integer">1976</yearofbirth>
</m:info>

</SOAP-ENV:BODY>
```

Complex and derived structures such as Linked lists, trees, queues, stacks, graphs and stacks can be serialized into SOAP encoding format using Linked Structures.

SOAP Arrays:

SOAP arrays are defined to contain the standard encoding type "SOAP-ENC: Array". Arrays can contain any kinds of elements or arrays. Arrays can hold either single reference or multi reference values. The accessibility and serialization of the array values depends upon the degree of reference.

```
float findTotal (int [] arrayData);

. . . . . .

float myData[5] = {100, 98, 78, 98, 98}
returnValue = findTotal (myData);
```

The above is a section of a C program where five numbers are stored in the array myData. The array myData is passed as parameter into the function findTotal. The function findTotal is expected to return a total of the five numbers.

The above method findTotal will be serialized into SOAP encoding format as:

```
<SOAP-ENV: Body xmlns:m="lathe.cims.rit.edu">

        <m: findTotal>
                <m:arrayData href="#array"/>
        </m: findTotal>

        <m:myData id="array" SOAP-ENC:arrayType="xsi:int[5]">
                <float>100</float>
                <float>98</float>
                <float>78</float>
                <float>98</float>
                <float>98</float>
        </m:myData>
```

</SOAP-ENV: Body>

If we look at the example above we can see that the array is reference independently using the **href/id** element. The Array element type must be or have to be derived from SOAP-ENC: Array.

SOAP Sparse Arrays:

SOAP specification provides support for transmitting sparse Arrays. Sparse Arrays are used when there is no need to transfer the entire array structure. These are used in situation where Arrays are allocated large bounds but do not contain enough elements. The position of the array index to be transmitted has to be indicated using "**SOAP-position**" attribute during serialization.

Partially Transmitted-Arrays:

SOAP provides support for transmitting partially transmitted arrays. Partially transmitted arrays vary in length.

```
<SOAP-ENC: Array; SOAP-ENC: arrayType="xsi: int[5]";
 SOAP-ENC:offset="[2]">

        <item>98</item>
        <item>100</item>

</SOAP-ENC:Array>
```

In the example above the array is transmitted starting at index position two which is indicated by the attribute "**SOAP-ENC-offset**". The "SOAP-ENC-offset" attribute is initialized to zero by default.

5.7. SOAP and Remote Method Invocation

5.7.1. Remote Procedure Call

Remote Procedure Call is the process of interacting between different processes. The notion of Remote Procedure Call has been there since the past two decades. Due to the advancement in networking infrastructure developers have looked for ways to distribute processes across multiple systems boundaries.

Remote Procedure Call can be compared to a C/C++ function call. When a call is made to a particular function by a program the caller pushes parameter onto the call stack. Then the execution control is transferred to the function. The function pops the parameters from the call stack, executes and puts the result back on the stack. The controls are then returned back to the caller. Remote Procedure Call is based on the same request/response

mechanism between caller and callee. RPC call mechanism is fairly simple when the processes lie in the same machine. It starts becoming complex when the procedures are distributed in different machines. Due to the advent of Object oriented programming RPC has to extend itself to the invoking of objects located in the same or different machines in the system.

SOAP is designed to be a replacement for binary RPC. SOAP is a lower level RPC Call and is not adopted enough to replace Distributed Object Computing like DCOM and CORBA completely. Technologies like DCOM and CORBA are built on top of Remote Procedure Call Architecture. However, vendors should be able to SOAP in their CORBA or DCOM implementation as a communication protocol. SOAP is a growing protocol and is very promising to send messages over the Internet.

5.7.2. SOAP and RPC

The following approaches are taken by SOAP in making Remote Procedure calls:

1) Identification of Remote Endpoints:

The most important step in establishing communication between two machines in the network is to open a connection on the remote machine. According to the TCP/IP specification this is achieved by mapping IP addresses and port number. DCOM and CORBA implement endpoint-mapping functionality, which maintains a table of interface-to-port pairs. When the connection is established between the two computers, data can be transported and methods invoked.

SOAP uses the same paradigm as DCOM and CORBA in identifying the Remote endpoint. The IP address of the HOST will be represented on the client's request packet. After the request is received and the task is carried out a response is sent back to the client.

2) Wire Representation of Data:

Distributed object technologies like DCOM (Distributed Component Object Module) are often known as *'COM on the wire'*. When the processes are located on the same machine inter-process communication is fairly simple. Since the processes are local to each other they can agree on the data format and the size of the parameters.

However, complexity increases when inter-process communication has to be established between different Computers in the network. To transmit or marshal parameters across the network the data has to be represented in a format, which can be interpreted equally at both the endpoints. To reduce this inconsistency Sun Microsystems introduced XDR (eXternal Data Representation). The XDR uses the big *endian order* representation where the most significant byte is placed at the lowest memory address. The XDR representation is specific to SUN Microsoft's hardware and works well on Java platform. Microsoft's DCOM and DCE RPC use Native Data Representation (NDR) in

representing data across the wire. Using NDR the Caller to write the code in receiving machine native data representation.

With distributed Object Model such as DCOM and CORBA the Interface Definition Language (IDL) is used to specify how the RPC call specific to the application should be represented on the wire. The completed IDL code is then compiled into a proxy/stub that is responsible for binding the application code to the networking code (Scribner and Stiver, 2000).

With SOAP the complexity of transmitting data on the wire is reduced. As we have already discussed HTTP and XML provide the protocol Transport and payload data representation for SOAP. XML is a plain text ASCII format that can be interpreted by any machine. Code has to be present at all the involved communication endpoint to parse and interpret SOAP message.

3) Interface Designs for Networks:

Remote Interface has to be designed carefully to minimize the number of round trip time between the communication endpoints. Each round trip can be costly and cost a lot of overhead. To avoid this problem it is a good idea to test remote interface during the system design phase. SOAP message has to be built carefully to optimize the network round trip time.

4) Passing By Reference:

When a caller calls a procedure there are to ways of passing the parameters. The parameters can either be passed by value or by reference. However, it is cheaper to pass parameters by reference than by value. Another reason to pass parameter by reference is to enable the function to modify the parameter. Within the same machine boundary any reference passed between callers to local procedure is valid as the procedures share the same address space.

However, passing references across machine boundaries becomes a difficult task. Technologies like CORBA and DCOM provide standard means of passing parameters and interface by reference across machine boundaries. However, passing parameters by reference becomes difficult in the case of SOAP because of the stateless nature of the protocol. State maintenance has to be done programmatically to make it easier to pass parameters by reference.

5.7.3. SOAP and ORPC

Remote Procedure call has evolved into execution of functions in distributed systems to invocation of objects. This is primarily due to the advent of object-oriented programming languages. The basic function of any remoting architecture is to convert a call stack into wire representation, to transmit data across network connection, and to reconstruct a call stack at the other end before executing the requested method (Scribner and Stiver, 2000).

Modern Distributed Object architectures like DCOM and CORBA interpret between call stacks and wire representation by the use of proxy. The remote endpoints and the data specification format are defined in the Interface Definition Language (IDL). The IDL is compiled to create proxy. **ORPC** (Object Remote Procedure Call) is a protocol, which DCOM uses to achieve wire remote object invocation. Similarly CORBA uses **IIOP** (Internet Inter-ORB Protocol) for remoting objects. DCOM and CORBA architecture serialize data into binary format and achieve remote method invocation.

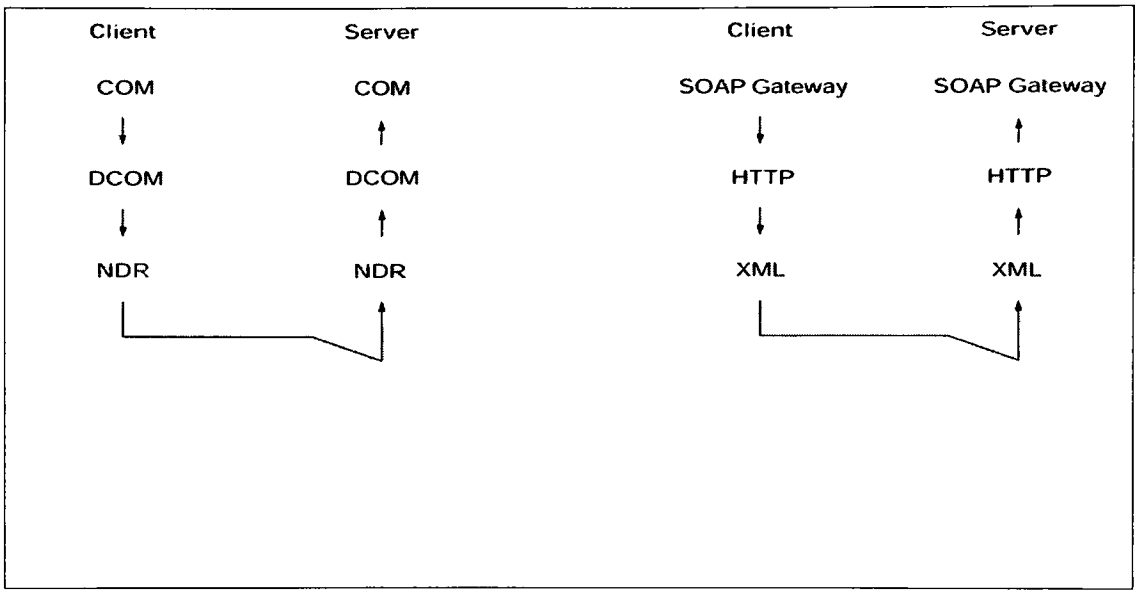

Figure 5.6: DCOM vs. SOAP (http://www.devx.com)

The SOAP architecture is similar to other Distributed Object Architecture as DCOM and CORBA. Figure 5.6, illustrates the difference between the approaches taken by DCOM and SOAP in remote method invocation. In case of DCOM/CORBA the data is converted into binary format using the IDL compiler and proxy/stub is created. The Proxy/stub have to be present in all the client and servers involved in communication. However, in case of SOAP data is serialized into XML payload before transmitting it on the wire. The SOAP Serializer acts as the proxy/stub as in the case of DOCM/CORBA. SOAP uses the HTTP protocol in transmitting data across the wire (as IIOP used by CORBA and ORPC used by DCOM).

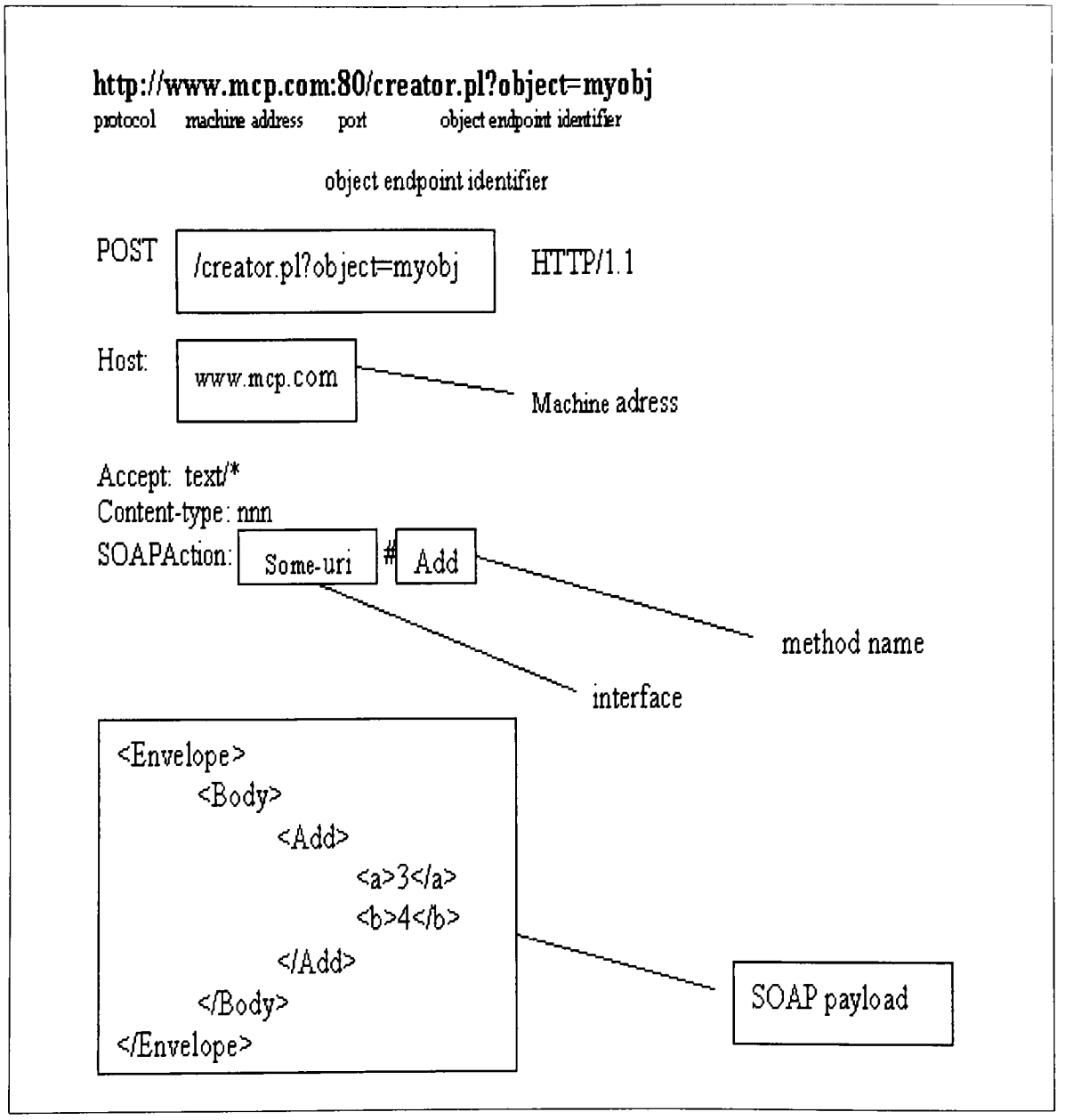

Figure 5.7, SOAP object Reference and Request (Scribner and Stiver, 2000)

Like any other Remoting architecture SOAP requires the following information in referencing objects located at remote endpoints:

1) Transport Protocol
2) Port Number
3) IP address
4) Object Identifier
5) Interface and method identifier

Figure 5.7, illustrates the constructs used by SOAP in referencing objects. The header of the SOAP architecture indicates that http has been used as the protocol. We can also see that port 80 has been used for communication. The ip 'www.mcp.com' has been used as the host IP address. We can that some-uri is used as the interface. Add is the name of the method to be invoked on the remote machine.

5.8. SOAP and its future

SOAP is a new technology and it requires time to gain acceptance. SOAP is not an implementation or anything as such. SOAP is a set of specification, which requires implementations to be accepted. SOAP is a technology, which is gaining acceptance slowly. All the major vendors as Microsoft, IBM and Sun Microsystems are adopting SOAP into their implementation.

SOAP is not a vendor specific technology. Microsoft was one of the first vendors to jump into this technology. SOAP has contributions from a group of developers and majority of big vendors like Microsoft, IBM and Userland Corporation. Any language, any platform and any developer can leverage SOAP technology.

SOAP as its name implies is simple. SOAP is designed to solve the complexities of distributed Object architecture in the business world. SOAP is a protocol, which uses xml extensively in its implementation. It is not a language-centric or platform-centric technology. SOAP message confides to a set of specification. Any language or platform with XML support should be able to interpret SOAP messages. SOAP is by design not an object-base transport. It was originally designed to be a purely RPC protocol. SOAP was originally designed to call a remote method and wait for a response. SOAP has a long way to go before it can support distributed Object technology fully. There have already been previous three drafts of SOAP specification with version 1.1 as the latest. The v 1.1 SOAP specifications have also enabled developers to create an Interface Definition Language for SOAP, which is called CDL. This interface makes use of XML schemas and the latest changes in XML technologies. SOAP is built in conjugation with extensible Markup Language (XML), which makes it extensible. The previous versions of SOAP specifications required HTTP to be used strictly as a transport for SOAP. However, the latest version of SOAP specifications, v 1.1 enables the use of other Internet protocols as SMTP and FTP.

We have to understand that SOAP has no direct support for remote object activation. Other technologies like CORBA and DCOM have direct support towards remote object invocation with an assumption that run time is present on both communication endpoints. SOAP is meant to be platform and language independent. SOAP is just a set of specifications. It is up to the developers to decide how to support and implement remote object invocation using SOAP depending upon the languages and platforms involved. Some vendors of CORBA technology have started using SOAP in their runtime environment.

SOAP seems to be very promising technology especially with the Internet. However, the success of SOAP depends how much it will be adopted and used by the Industry. It is a growing protocol and requires major work and support from the developer. Microsoft was one of the first vendors to jump into SOAP. SOAP has gained special attention after becoming accepted by IBM. Now, everyone in the industry is looking towards the premises of SOAP.

Now that SOAP has been subsumed into the XML Protocol work, the big vendors have (for the most part) stopped arguing about SOAP and we have a fairly open process for beating the protocol into shape (http://www.xml.com/pub/a/2001/04/04/soap.html). The XML protocol Activity group is a part of W3C that focuses upon promoting communication in distributed environment using XML as its encapsulation language. So, the next draft of SOAP (not Dave Winer's SOAP version 1.2) is yet to be released by the XML Protocol Activity group at the time of this writing.

References

"Microsoft Biztalk Sever 2000: Documented (Pro-Documentation)" Microsoft Corporation. Washington: Microsoft Press.

"SOAP: Simple Object Access Protocol". Microsoft Corporation. <**http://msdn.microsoft.com/xml/general/soapspec.asp**>.

"When SOAP a Good idea in a project" Ejbinfo.com. <**http://ejbinfo.com/articles/00/10/28/0933234.shtml**>.
"SOAP Zone" VBXML.COM. < http://www.vbxml.com/soap/>.

Brown, Kent. "SOAP for Platform-Neutral Interoperability". <**http://www.xmlmag.com/upload/free/features/xml/2000/04fal00/kb0004/kb0004.asp**>.

Box, Don. "A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages" <**http://msdn.microsoft.com/msdnmag/issues/0300/soap/soap.asp**>.

Borders, William. 'SOAP: Simple Object Access Protocol' <**http://idm.internet.com/articles**>.

Cartwright, Adam S. Soap Soup. < http://www.xml101.com/articles/adam/soapsoup/default.asp>.

Chartier, Robert. "SOAP: Cleaning up you Developer Link" <**http://www.aspfree.com/authors/robert/view.asp?aid=29**>.

Chartier, Robert. "Sample SOAP implementation". <**http://www.aspfree.com/authors/robert/view.asp?aid=19**>.

Dodds, Leigh. "Converging Protocols" < **http://www.xml.com/pub/a/2000/12/20/ebXML.html**>.

Egan, Syd. "Simple Object Access Protocol: A Step-By-Step Approach" <**http://www.vbip.com/xml/soap_syd.asp**>.

Esposito, Dino. "Simulate SOAP and Web Services" <http://www.xmlmag.com/upload/free/features/xml/2000/05win00/de0005/de0005.asp>.

Gallagher, Sean. "SOAP, BizTalk, and Super Scalability" <**http://www.xmlmag.com/upload/free/features/xml/2000/02spr00/evspr00/ev1spr00.asp**>.

Jazdzewski, Chuck. 'What Operating System who cares?'. <**http://www.futureofsoftware.net/cj0010/cj0010.asp**>.

Marcato, David. "Distributed Computing with SOAP" <**http://www.devx.com/upload/free/features/vcdj/2000/04apr00/dm0400/dm0400-1.asp**>

Ogbuji, Uche. "XML messaging with SOAP" < **http://www-105.ibm.com**>

Scribner, Kennard. Understanding SOAP: The Authorative Solution. Indiana: Same Publishing, 2000.

Shohoud, Yasser. "Creating Firewall-Friendly Distributed Apps" <**http://www.devx.com/upload/free/features/vbpj/2000/10oct00/ys0010/ys0010.asp**>.

Travis E., Brian. XML and SOAP Programming for Biztalk servers. Washington: Microsoft Press 2000.

Winer, Dave. "Dave's History Of SOAP" < **http://www.xmlrpc.com/discuss/msgReader$555**> (September 25, 1999).

Winer, Dave. "SOAP meets RSS" <http://www.thetwowayweb.com/soapMeetsRss>

Winer, Dave. "RPC over HTTP via XML" <**http://davenet.userland.com/1998/02/27/rpcOverHttpViaXml**> (Feb 27, 1998).

Winer, Dave. "A busy developers Guide to SOAP 1.1' <**http://www.soapware.org/bdg**> (April 02, 2001).

Woodgate, Scott. Professional Biztalk Programming. Birminghan: WROX Press, 2000.

6. AsiStore Shopping Cart Application

6.1. Introduction to AsiStore Shopping Cart Application

Active Server Pages (ASP) is a server-side scripting tool available on Microsoft's Internet Information Server (IIS) platform. Active Server Pages extends the power of conventional scripting languages as CGI, JavaScript and Pearl by utilizing the features of the server. Components written in fourth Generation programming languages as C++, Visual Basic and Delphi can be instantiated on the server side by using Active Server Pages scripting. Active Server Pages revolutionizes the WWW by providing full support to reusable components, extensible Markup Language (XML), dynamic HTML and server side scripting. Competitors of Active Server Pages technology includes SUN Microsystems's Java Server Pages (JSP) another server-side scripting utility based on Java.

Application running under Active Server Pages can run under any Web Browser as all the processing is carried out on the server side and only the resulting HTML contents are displayed on the client's browser. Active Server Pages offers the power to utilize the latest features of the component technology and any ODBC (Open Database Connectivity) compliant databases.

The goal of this shopping Cart Application is to demonstrate Information Sharing making use of Internet technology such as XML, HTTP and Simple Object Access Protocol. In this application we serialize our Remote Method invocation call into SOAP messaging format. Then we transfer the XML document to the remote Machine by using the Hyper Text Transport Protocol. We make use of a SOAP specific architecture to establish communications in between remote objects.

6.2. Overview Of SOAP-XML Driven Shopping cart Application

We use AsiStore website as an application to illustrate the architecture used and concepts of XML and SOAP. AsiStore is a virtual Computer store that sells hardware and software products. It does business with two other stores, Store1 and Store2. Store1 manufactures and supplies computer hardware products. Store2 is a distributor of Computer Software. These stores maintain their inventory. AsiStore website is a virtual store where customers interested in computer products come to.

Each potential customer requests a free membership to the AsiStore. An username and password is assigned to each potential shopper. Once a customer is logged on and authorized, a catalogue (Figure 6.7) of available products (from Store1 and Store2) is presented. The customer is not aware of and is not concerned of where the products are coming from when He/She is shopping at the AsiStore. A virtual shopping cart is created for customer selected item (Figure 6.8) and is modified by by adding /removing items to/from the shopping cart. An order is submitted when the shopping is done. The billing/credit card information is then collected (Figure 6.10). An order number is generated once all the information is entered and accepted.

This application is implemented using XML and SOAP concepts. The data stored at the stores use XML structure and the communications among the stores, Store1 and Store2 use SOAP protocol. The implementation detail is described in Figure 6.1.

6.3. Implementation Details

At the time of writing the version of the software used in developing this application includes:

1) Microsoft Visual Studio 6 package.
2) Microsoft XML parser 3.0
3) Internet Information Server 5.0
4) XML specification version 1.0 as defined by W3C consortium
5) SOAP Specifications 1.1 as defined by W3C consortium

Before we discuss the intricacies of the SOAP-XML Driven Shopping cart Application we will outline the process when a user enters the website. Figure 6.1 shows the overall architecture of the systems involved with the application. These steps below are required to implement the application as described:

1) Display the store Front.
2) Authorize/Register User and check Store Availability.
3) Retrieve data from Store1 and Store2.
4) Display the list of available products.
5) Fill Shopping Cart.
6) Obtain Shipping/Billing information.
7) Confirm Order.

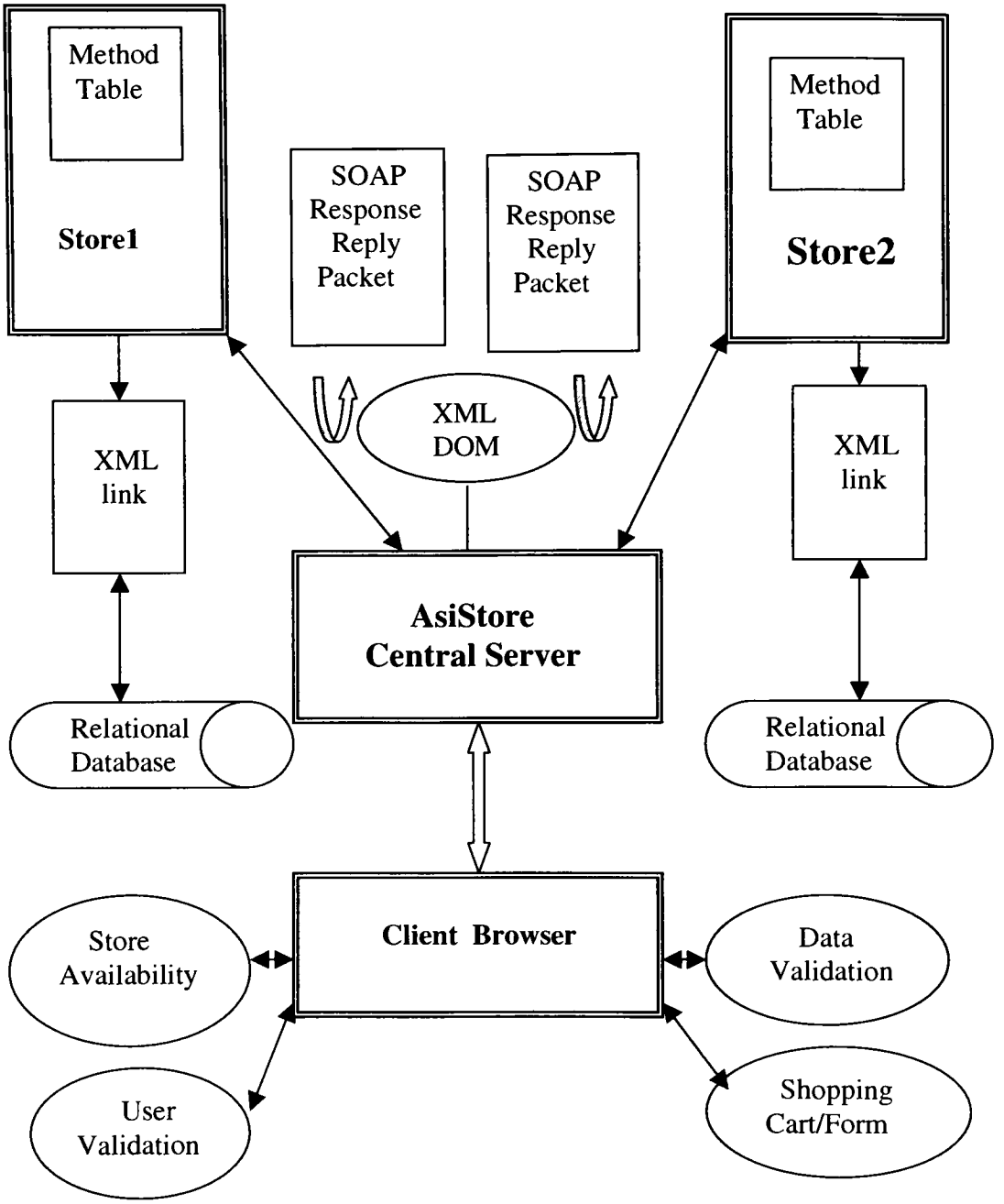

Figure 6.1: Software Architecture for Application

The complete set of code used in this application is available in the appendix. We will go through each of the step listed above and functionalities associated with each step. The logic and workflow for the application are shown in figure 6.2. The transaction starts with the index.asp page. The user will have to login to the memberlogin.asp page to shop.

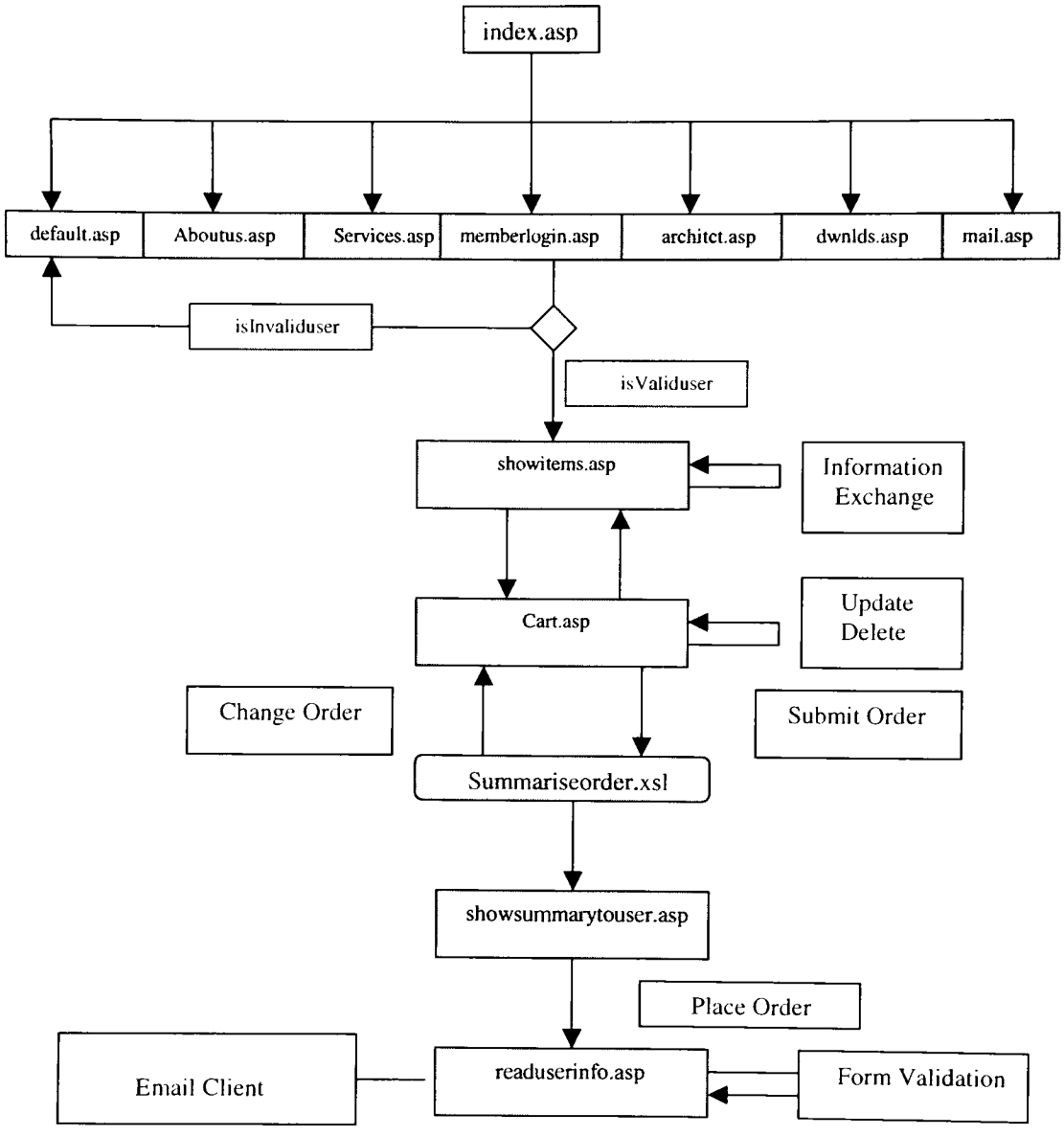

Figure 6.2: Application Overflow Diagram.

1) Display Store Front.

The AsiStore shoppers will access the store using a web browser (e.g. Microsoft Internet Explorer, Netscape Navigator) and the specified URL. A welcome screen invites the users to the screen.

The application will initially display the index page index.asp in the browser. This page uses two HTML frames driven by menu.asp and default.asp. Figure 6.3 shows the index.asp displayed to any user visiting the AsiStore by default. The menu.asp page contains the list of pages the user will be able to surf into. One of the choices on the menu bar is Member Area. Each user has a Session limit to approximately twenty minutes.

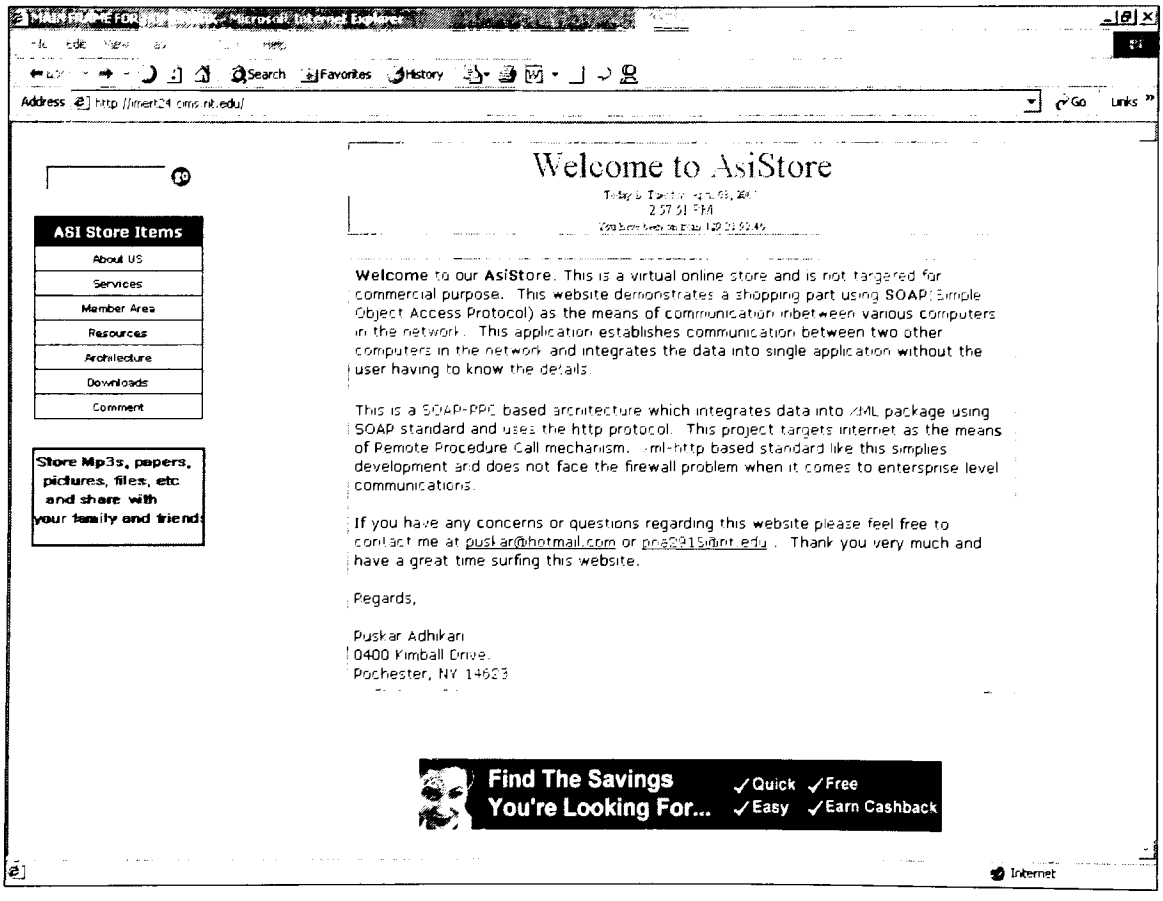

Figure 6.3: Index.asp page showing the menu items and the welcome page.

2) Authorize/Register User and check Store Availability.

Existing members will login with their username and password while new users can become members from this screen. Availability of merchandize from the two stores is verified.

The user is required to provide username and password after being directed to the memberlogin.asp page (Figure 6.4). After the user provides the login name and the password then the function isLegaluser (txtusername, txtpassword) is executed which verifies that the user is an authorized member of the AsiStore Shopping community. The membership is maintained for product update information and keeping record of each user transaction. Customers can register to become member of AsiStore by clicking on the New Users button.

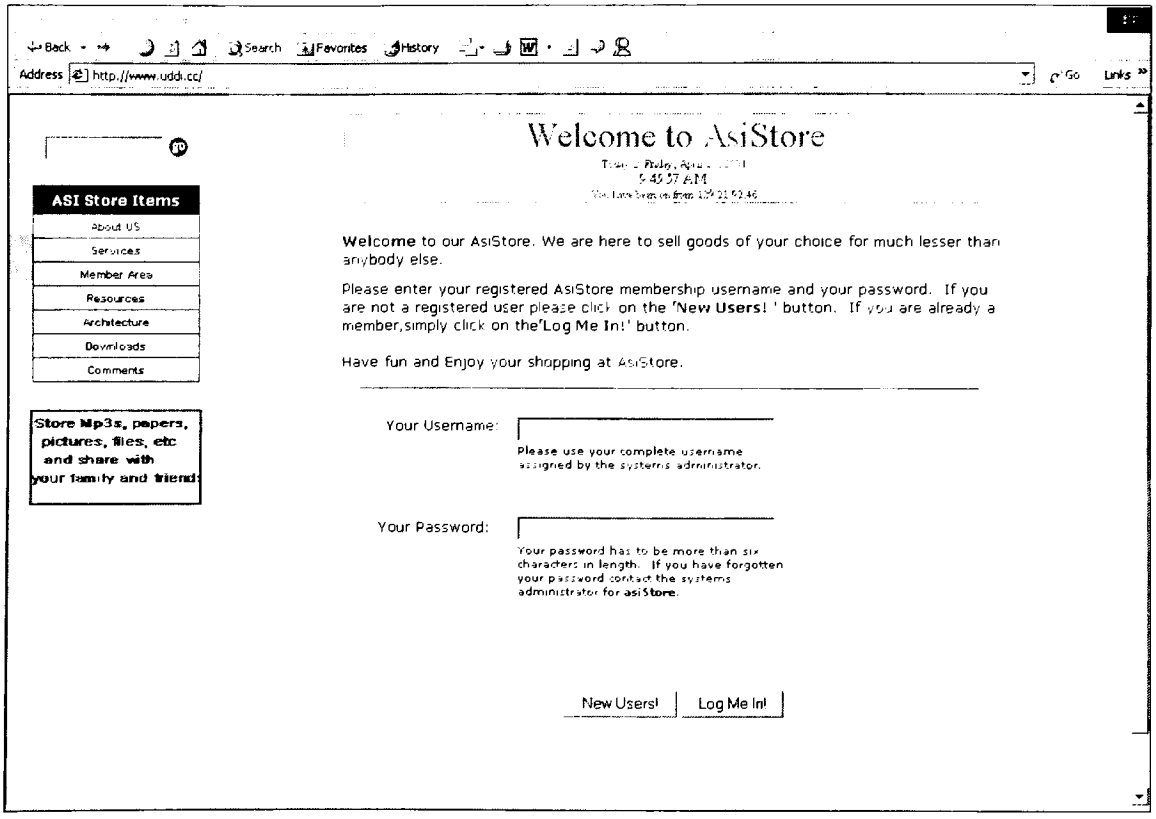

Figure 6.4: User login page.

The function is isLegaluser scans the database table usertable and checks for the availability of the user entered username and password. If the user is listed in the database table a Boolean value of one will be returned by the function and Boolean value of zero otherwise. If the user is a valid member of according to the value returned by the function isLegaluser then the user will direct to the page redirector.asp. Then the central Server will ping the servers residing on Store1 and Store2. Making a call to the

pingMachine function will carry out this ping operation. This function pingMachine takes in the ip addresses of the remote server as a parameter and pings to see if the machine is up and running on the network. If the Ping is successful then function will return a value of zero. If the ping is unsuccessful then items will not be available from that particular store.

4) Retrieve data from Store1 and Store2.

The application will retrieve a list of products available and/their pricing data for display and selection.

After the ping is successful the component **pullxmlfromstring** will be instantiated on the AsiStore Server (redirector.asp). The **pullxmlfromstring** component is written as ActiveX DLL using Visual Basic 6. Then an xml String template (<Products></Products>) is loaded onto the Document Object (DOM) tree, which creates an in-memory representation. The Document object Module object is initialized as a session object to manage state during the entire application session. This DOM tree will contain the ProductID, ProductName, Units and Price of the items and is used in driving the shopping cart, which is displayed as the page **cart.asp** (Figure shopcart) on the browser.

The interfaces provided by the **pullxmlfromstring** component used in manipulating the DOM tree are:

*1) **Public Function Load_String (xmlString)**: This function will take in a String template as parameter and initialize the XML DOM tree.*

*2) **Public Function addItem (pid As Variant, productsname As Variant, productsprice As Variant, productunits As Variant)**: This function will add the individual items into the XML tree.*

*3) **Function getItem (pid As Variant)**: This function checks to see if a particular item is already on the XML tree.*

*4) **Function updateItem (pid As Variant, productsunits As Variant)**: This function updates a particular item in t he XML tree.*

*5) **Function deleteItem (pid As Variant)**: This function will delete the specified item from the XML tree.*

*6) **returnNumunits (pid As Variant)**: This function returns the number of particular item available from the XML tree.*

*7) **givexmlforDisplay ()**: This function dumps the returns the XML tree as a String.*

After the DOM tree is loaded into the memory, the user will be redirected to the page *showitems.asp.* During the redirection period to the page (showitems.asp) the central server (AsiStore) communicates with the servers associated with the Store1 and Store2 if they are available. The central Server will make a call to the function *communicatewithServer* to send and receive SOAP messages. This function *communicatewithServer* takes in the URL of the store to communicate as parameter.

This function *communicatewithServer* will send SOAP message to Store1 and Store2 in the following format:

SOAP REQUEST PACKET

POST: HTTP:/1.1
HOST: storedestination
Content-Type: text/xml

<SOAP-ENV: Envelope xmlns: SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:m="http://lathe.cims.rit.edu/example/"
SOAP-ENV:encodingStyle=http://schemas.xmlsoap.org/encoding/>

<SOAP-ENV: Body>
<m:returnxmlstring>
.....(Parameters)
</m:returnxmlstring>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

At each of the available stores the component **scanadotoxml** will be instantiated when a SOAP message is received from AsiStore. The **scanadotoxml** component is written as ActiveX DLL using Visual Basic6 and is registered on all the machines, Store1 and Store2. ActiveX DLL's are registered using the command *regsvr32 filename.dll* from the windows command prompt.

The **scanadotoxml** component converts the product storage database table items into the AsiStore data exchange standard as shown in figure5.

```
<?xml version="1.0"?>

<Products>
  <Product>
        <ProductID>.....</ProductID>
              <ProductName>.....</ProductName>
              <Units>.....</Units>
              <Price>.....</Price>
  </Product>
. . . . . .
</Products>
```

The node definitions are:

1) Products: *root note of the XML document*

2) Product*: First child to the root node Products.*

3) ProductID: *First Child of the Product Node.*

4) ProductName: *Second Child to the Product Node.*

5) Units: *Third Child to the Product Node*

6) Price: *Fourth Child to the Product Node*

Figure 6.5: AsiStore data exchange Standard.

The resulting xml string which confides with the AsiStore data exchange standard will be wrapped into a SOAP Response message and sent back to the AsiStore Server. A Successful SOAP Response from one of the stores would look like:

SOAP RESPONSE PACKET

POST: HTTP:/1.1 200 OK
Content-Type: text/xml

<SOAP-ENV: Envelope SOAP-ENV:
SOAP="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:m="http://lathe.cims.rit.edu/example/"
SOAP-ENV:encodingStyle=http://schemas.xmlsoap.org/encoding/>
<SOAP-ENV: Body>
<m:returnxmlstringresponse>
(embedded result in xml string)
</m:returnxmlstringresponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

If the SOAP message is not successfully invoked then a fault message indicating the failure will be sent back to the AsiStore, which might look like:

SOAP FAULT Message Packet

POST: HTTP:/1.1 500 Internal Server Error
Content-Type: text/xml

<SOAP: Envelope xmlns: SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-EVN:m="http://lathe.cims.rit.edu/example/"
SOAP-
ENV:encodingSTYLE="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV: Body>
<SOAP-ENV:Fault>
<faultcode> SOAP:Server.InternalError</faultcode>
<faultstring>
Internal Server Error
</faultstring>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

The SOAP message may fail due to certain HTTP and network errors. The different possible HTTP response Status Codes that could be indicated in a SOAP message are:

1) ***200 OK*** [SOAP Successful Response]
-Request succeeded

2) ***301 Moved Permanently*** [SOAP Fault Code]
-request object moved, new location specified [SOAP faultstring]

3) ***400 Bad Request*** [SOAP Fault Code]
-request message not understood by server [SOAP faultstring]

4) ***404 Not Found*** [SOAP Fault Code]
-Request document not found on this server [SOAP faultstring]

5) ***505 HTTP version not supported*** [SOAP Fault Code]

Fault can be caused by Network problems, machine downtimes, memory leaks, and high volume of traffic and http errors. If a SOAP Fault Message is received then items will not be listed on the web browser from that particular store (Store1, Store2).

Figure 6 shows the overall algorithmic approach used in sending and receiving SOAP messages in-between two machines in the network. The outline of the algorithm associated with the function communicatewithServer function includes:

1) The Caller creates a Request payload in XML format (soap) and includes the name of the method to invoke and the required parameters.

3) The XML Payload is wrapped into HTTP POST command packet.

4) The wrapped SOAP message is sent to the remote machine using the Http protocol and the ipaddress. (Store1 and Store2)

5) Then the caller waits for a reply from the remote machine.

6) Remote machine (callee) receives the message and de-serializes the message

7) It identifies the wrapped methods and invokes it.

8) If successful the callee will create a XML payload including the values returned by the method. Incase of failure it will create a SOAP FAULT Response message.

9) Then the remote machine (callee) will send the message back to the caller.

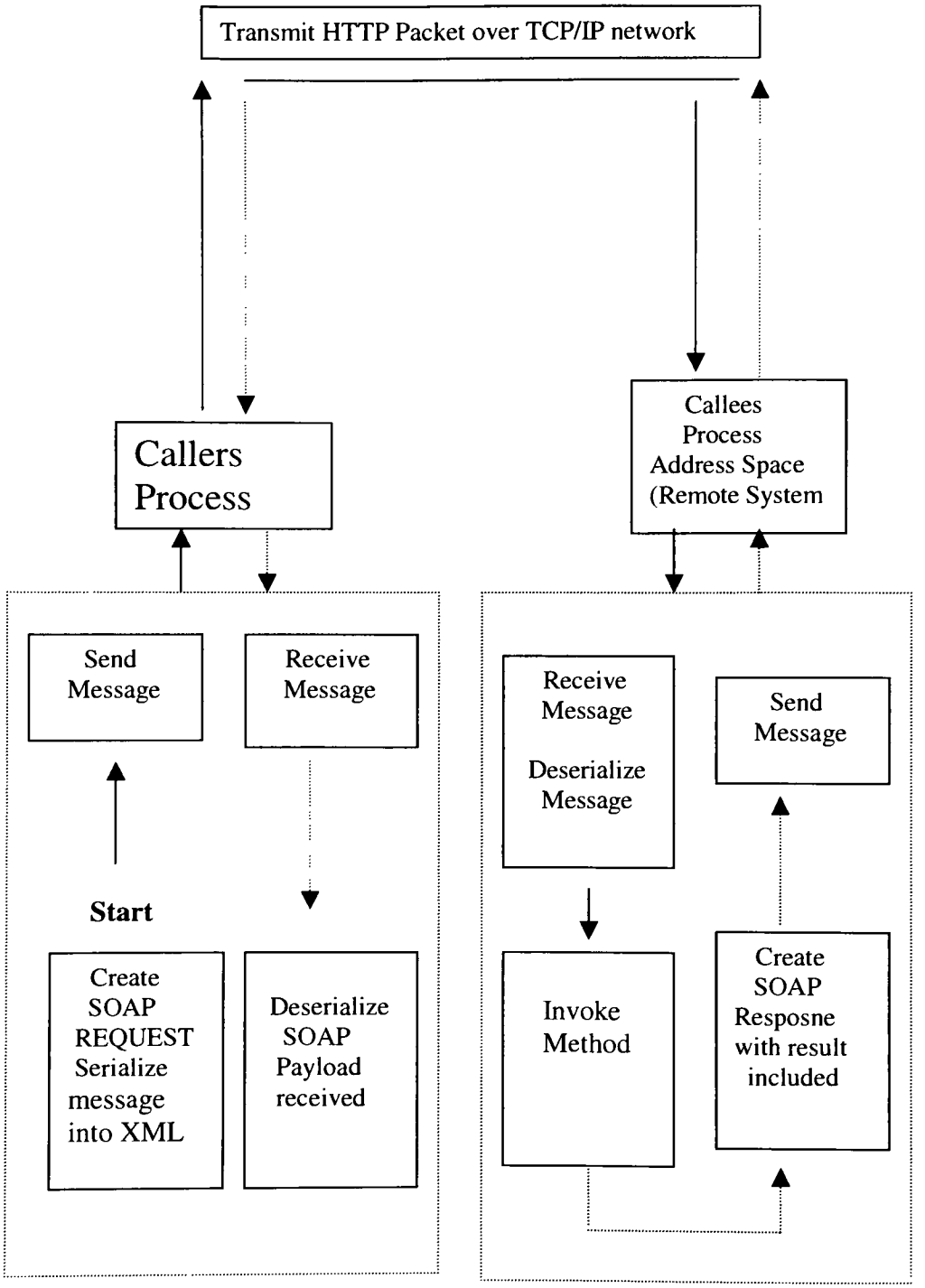

Figure 6.6: Caller/Calle Messaging architecture

5) Display the List of Available Products.

The AsiStore shoppers will be displayed with a list of item items and pricing information on the web browser as shown in Figure 6.7. AsiStore shoppers will be able to view their shopping basket by clicking on the view cart button at the bottom of the page. The shoppers will also be able to select item from the list by clicking on the Add Item button.

If the messages received from the stores are successful the XML string containing the product information will be extracted from the SOAP Response. The product information extracted from each of the stores available is combined into a single XML string. Then the function showlistitems is called and the resulting XML string from the available stores is passed as a parameter to the function. The function showlistiems will return a HTML table containing the list of items available from to the user. Then the showitems.asp page will display the resulting HTML string to the browser as shown in figure 7. The user will be able to select items available from the catalogue.

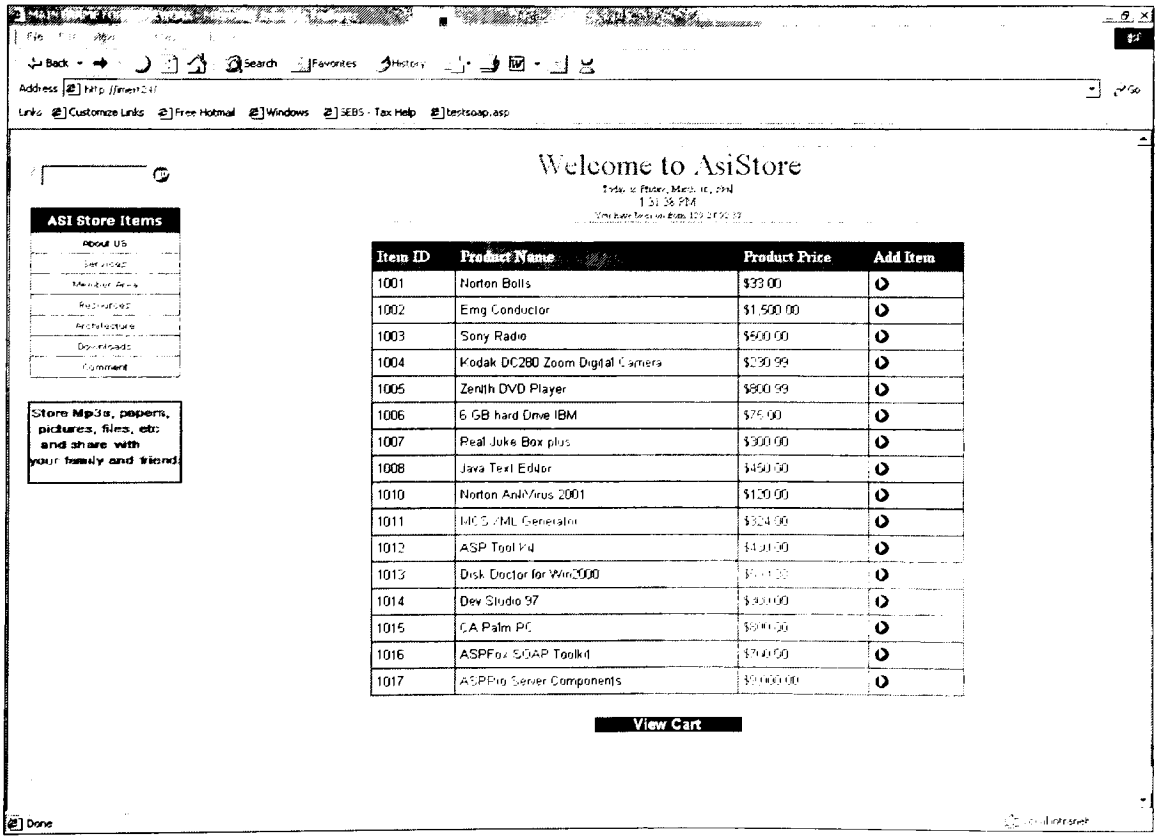

Figure 6.7. List of Items available at AsiStore.

6) Fill Shopping Cart.

AsiStore shoppers will be able to add items onto the shopping basket from the list of products. The shopping basket will contain the list of items, which have been selected by the AsiStore shopper. The shopping basket will also contain units and running totals of the items purchased. The shopper has the option of deleting existing items, updating the units of item purchased, and going back to view the product listings. The Shopper can view the summary of items purchased by hitting submit order button.

When a particular item is selected, the item will be added to the shopping basket. The shopper is directed to the page cart.asp upon the selection of an item. The shopper is able to go back and forth between the page showitems.asp and cart.asp during the shopping session. The user is able to modify delete or update items from the shopping cart. A picture of a live shopping cart used in Customer transaction is shown in figure 6.8. The interfaces from the **pullxmlfromstring** component are used by the application in manipulating items in the Shopping Cart. The application makes a call to the function showshoppingbasket, which returns the shopping basket and its contents as a HTML string. The resulting HTML string is displayed on the browser.

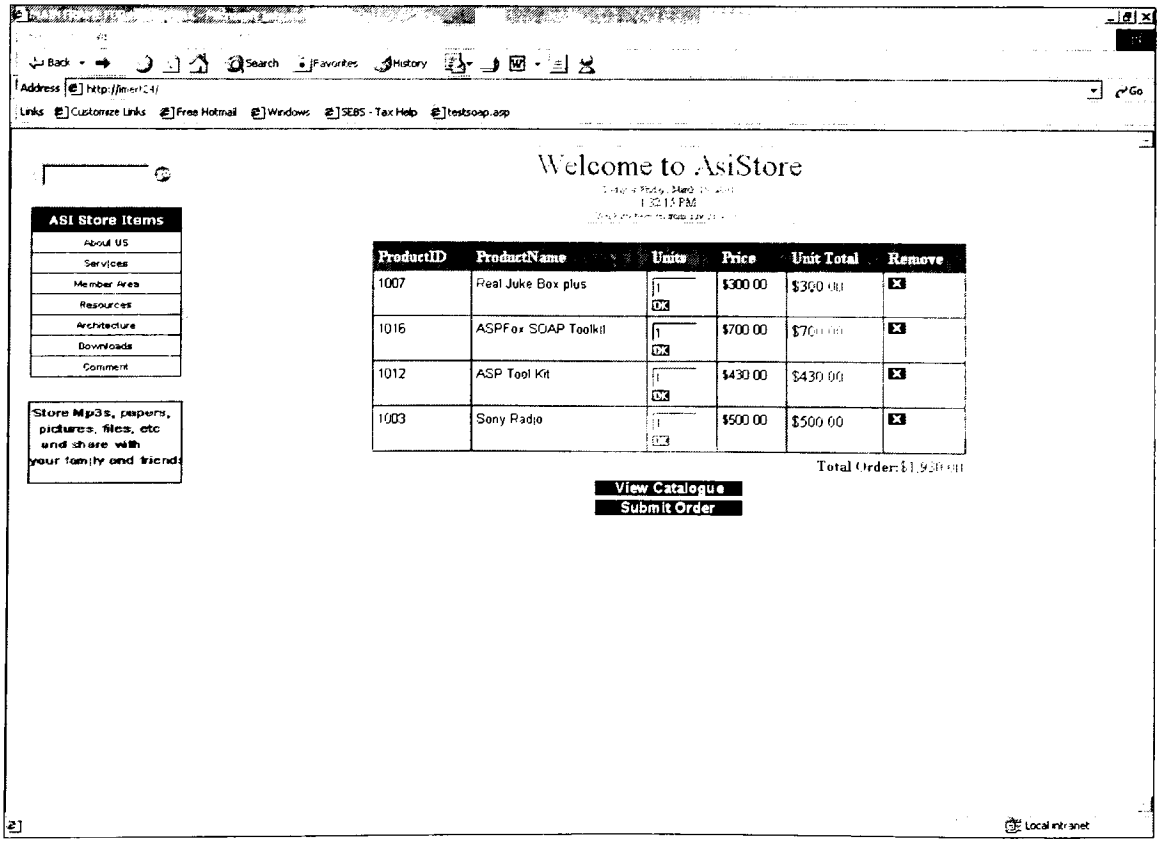

Figure 6.8. Picture of the shopping cart during transaction.

After the order is submitted a page summarizing the list of items selected (showsummarytouser.asp) is displayed on the browser. This page uses XML Stylesheet (XSL) to present the information to the shopper. The shopper will still have the option of going back and changing order at this point of the application. The page showsummarytouser.asp transfers the items in the shopping cart (XML tree) into HTML using the XSL sheet summariseorder.xsl. A sample summary page is shown in figure9. Java scripting is used in inside the XSL sheet summariseorder.xsl to calculate the Unit and the running Totals.

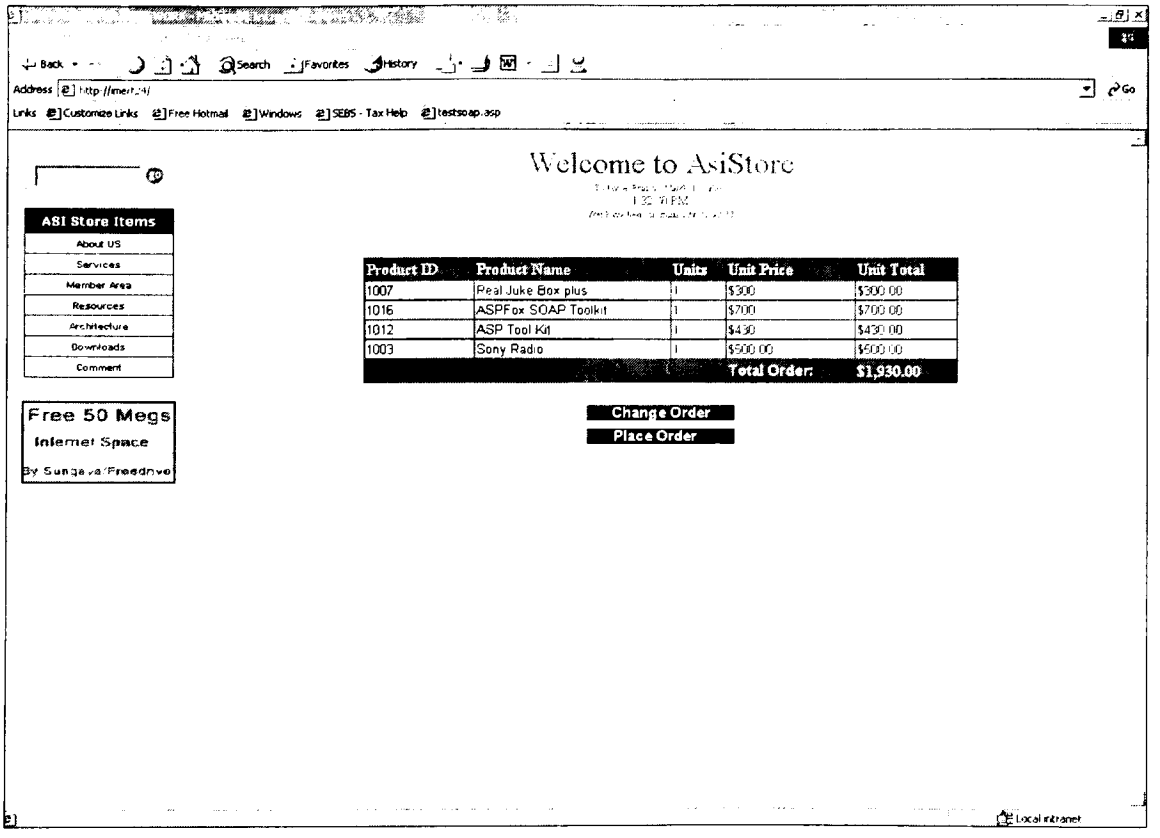

Figure 6.9. Page showing the summary of items purchased.

7) Obtain Shipping/Billing information.

Upon the final Submission of order the shopper is required to fill in the shipping and the billing information. The required information has to be filled correctly for prompt delivery.

The user will then be asked fill out the shipping and handling information to confirm the order. This will be carried out by the page readuserinfo.asp. A picture of the form, which the user will be required to fill out, will look like in figure 6.10. The application uses Server side validation to make sure the information is filled out correctly. Regular expression validation is used to authenticate password and Credit card information.

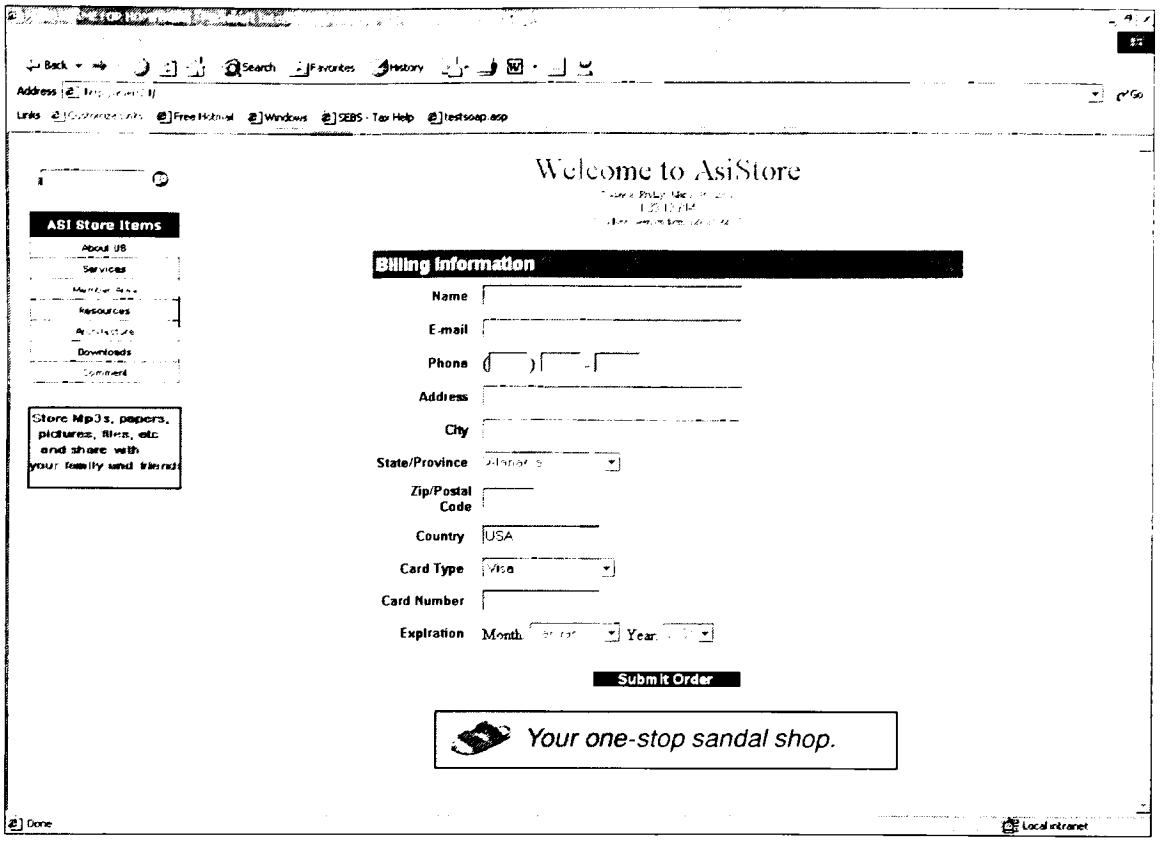

Figure 6.10. Billing and Credit information form.

8) Conform Order.

When the order is submitted conformation e-mail will be sent to the user containing a transaction ID and the address information.

After the required information is filled out and is valid a confirmation page will be displayed to the user. Email will be sent to the User's email address. The application calls the function sendemail to send conformation email to the AsiStore Shopper. The sendemail function uses the SMTP server and CDNOTS object available with the IIS 5.0. The final transaction will be save into database tables using the function **saveuserTransaction**. The database tables used in this application are described in the preceding section.

6.4. Database Structure

6.4.1. Store1 and Store2 database table Structures

Visual FoxPro database tables were used in the shopping cart application. The Stores Store1 and Store2 used different table structures in storing product information. The database tables are programmatically transferred to fit into the AsiStore data exchange Standard (Figure 6.5) before finally being transferred using SOAP messaging.

Store1 table:

Store1 stored its product information in the database table product1. The structure for the table store1 is shown in the figure below (Figure 6.11).

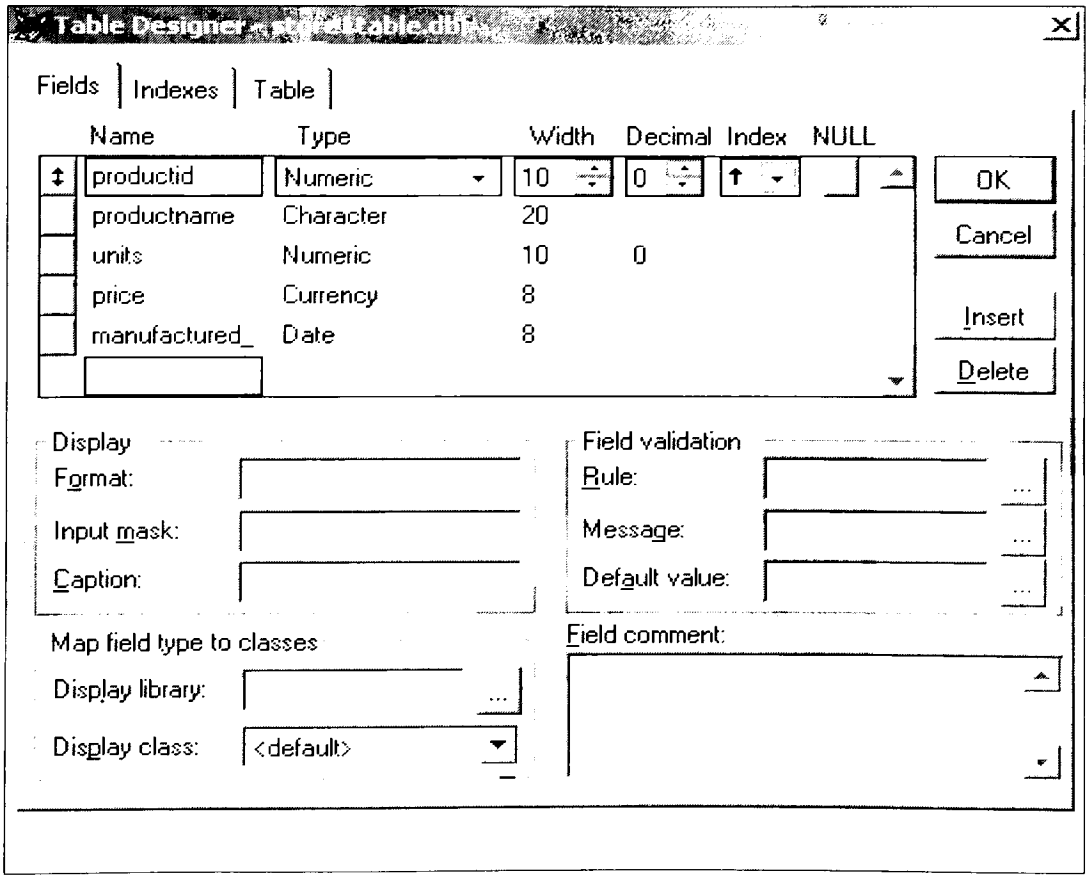

Figure 6.11. Structure of the table maintained by Store1.

Store2 Table:

Store2 stored its product information in the database table product2. The structure of the table store2 is shown in the figure below.

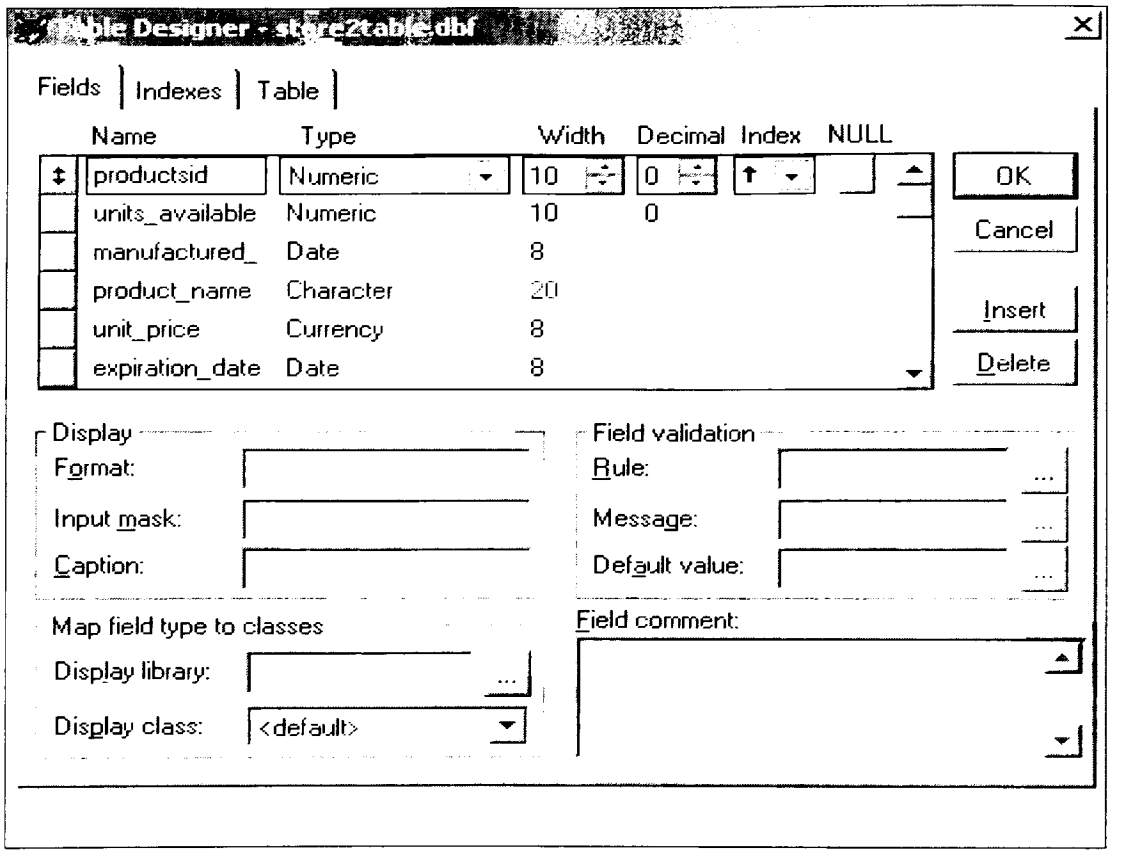

Figure 6.12: Structure of the table maintained by store2.

6.4.2. AsiStore Table Structures

The AsiStore maintains a table with list of usernames and passwords used in user authentication.. The AsiStore also maintains a record of all the transactions committed by users for future use and delivery of products on time.

Usertable table:

The table usertable.dbf (figure 6.13) stores the login information for the users and is maintained by the AsiStore. This table contains the username, password and userid for each registered users. The userid and username name form a composite key and identifies each user uniquely. The userid is auto incremented when a new user is added.

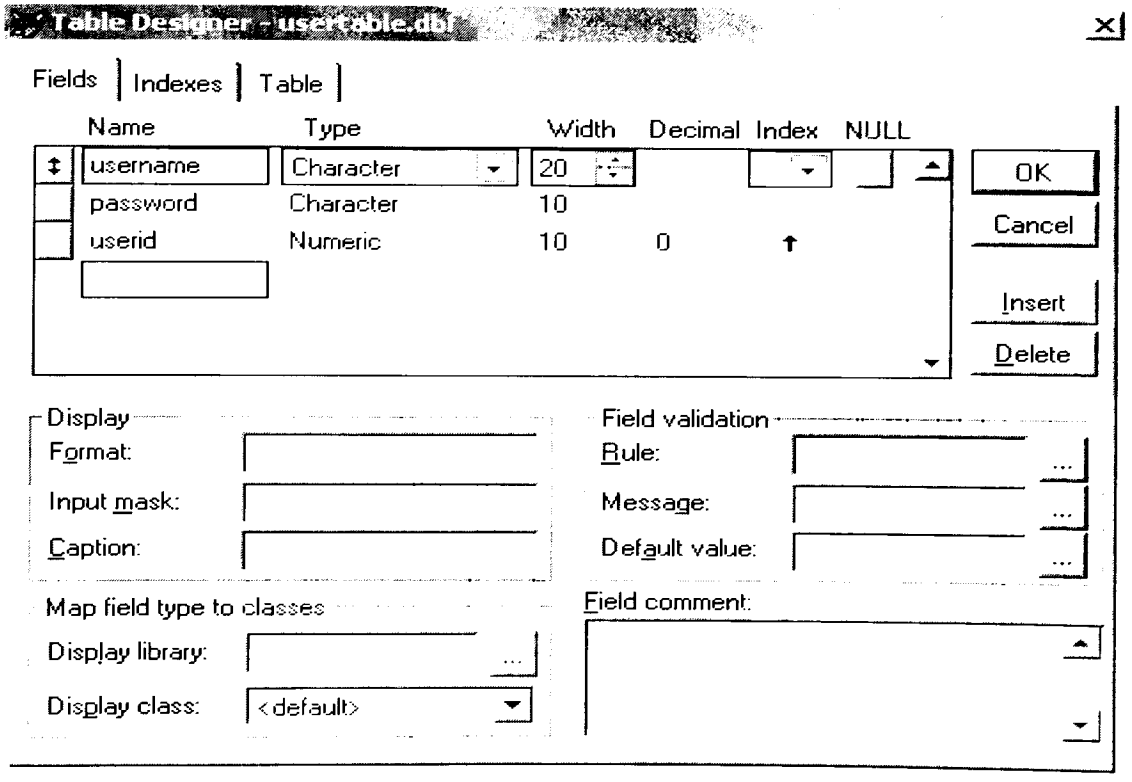

Figure 6.13: Table used for maintaining user authentication information.

The AsiStore maintains three sets of table userinfo.dbf, transactionrecord.dbf and items.dbf in keeping tract of each of the transaction committed by users. The Figure below (Figure 6.14) illustrates a relationship maintained by the tables in storing transaction information.

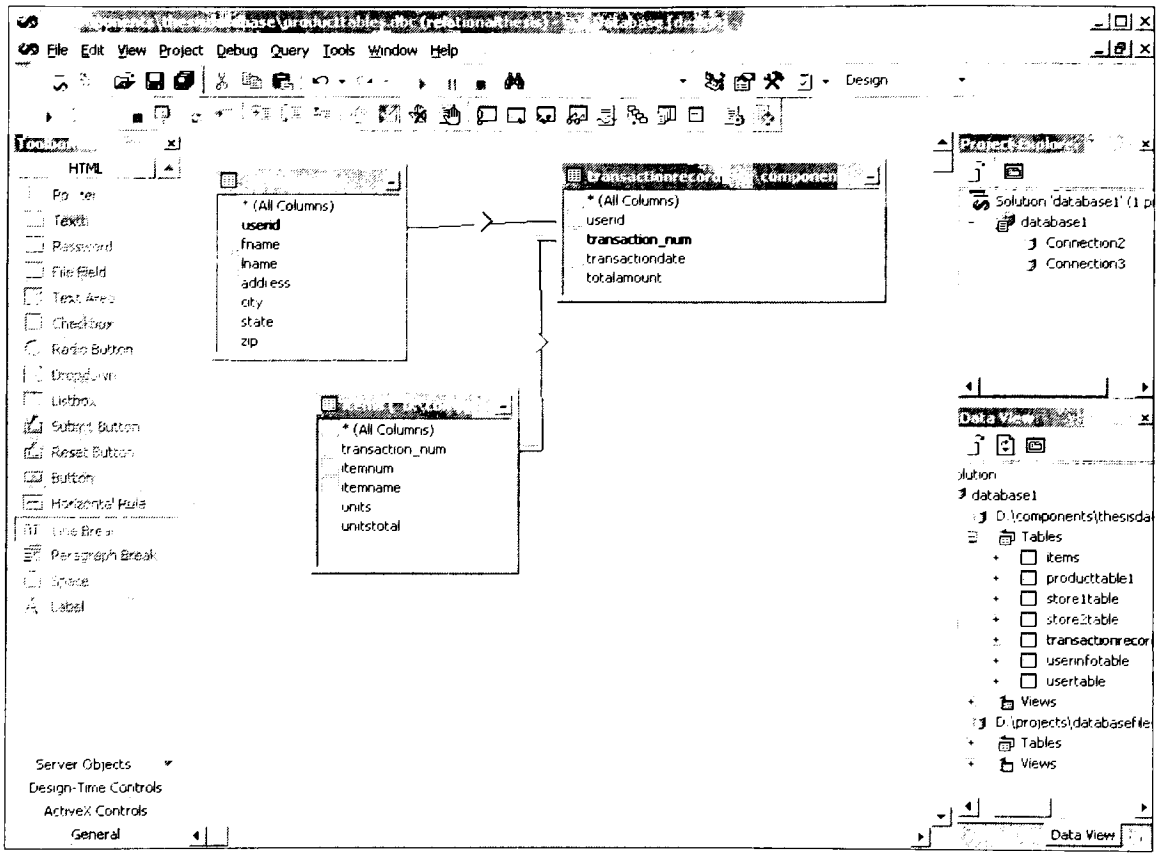

Figure 6.14: illustrating relationship between tables involved in storing transactions committed by users.

Userinfo table:

The useinfo table stores the billing and shipping information of the user. It uses the field userid as the primary key. The userinformation table structure will look like in Figure 6.15 below.

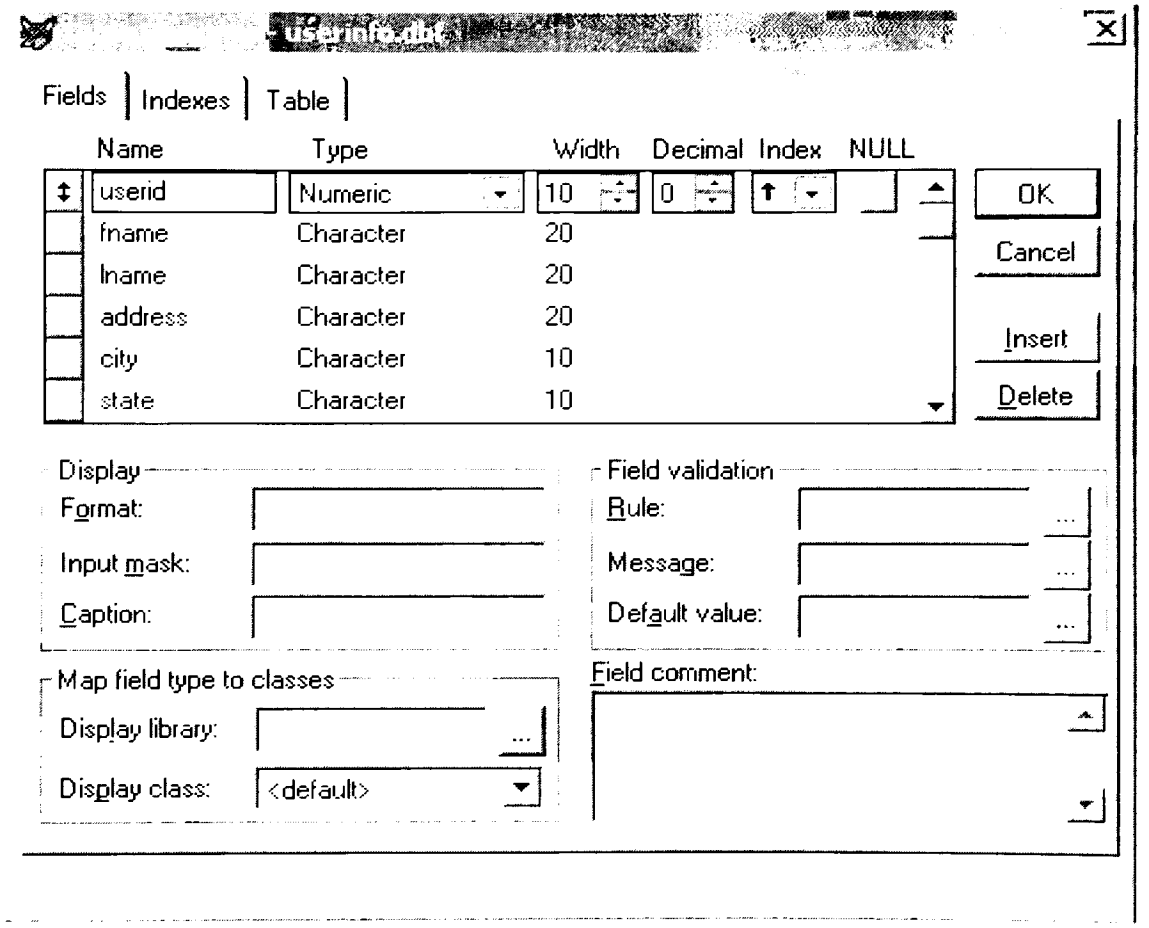

Figure 6.15: Table structure for Userinfo table.

Transactionrecord table:

The transaction record table is used in storing the transaction information committed by users. The field userid, which is a primary key in the table userinfo, is used as a foreign key in this table. Userid is used in identifying the user for each transaction committed. The field transaction_num is defined to be the primary key in this table. Each transaction_num is unique and generated automatically for every transaction. The structure of the table transactionrecord.dbf is illustrated in Figure 6.16 below.

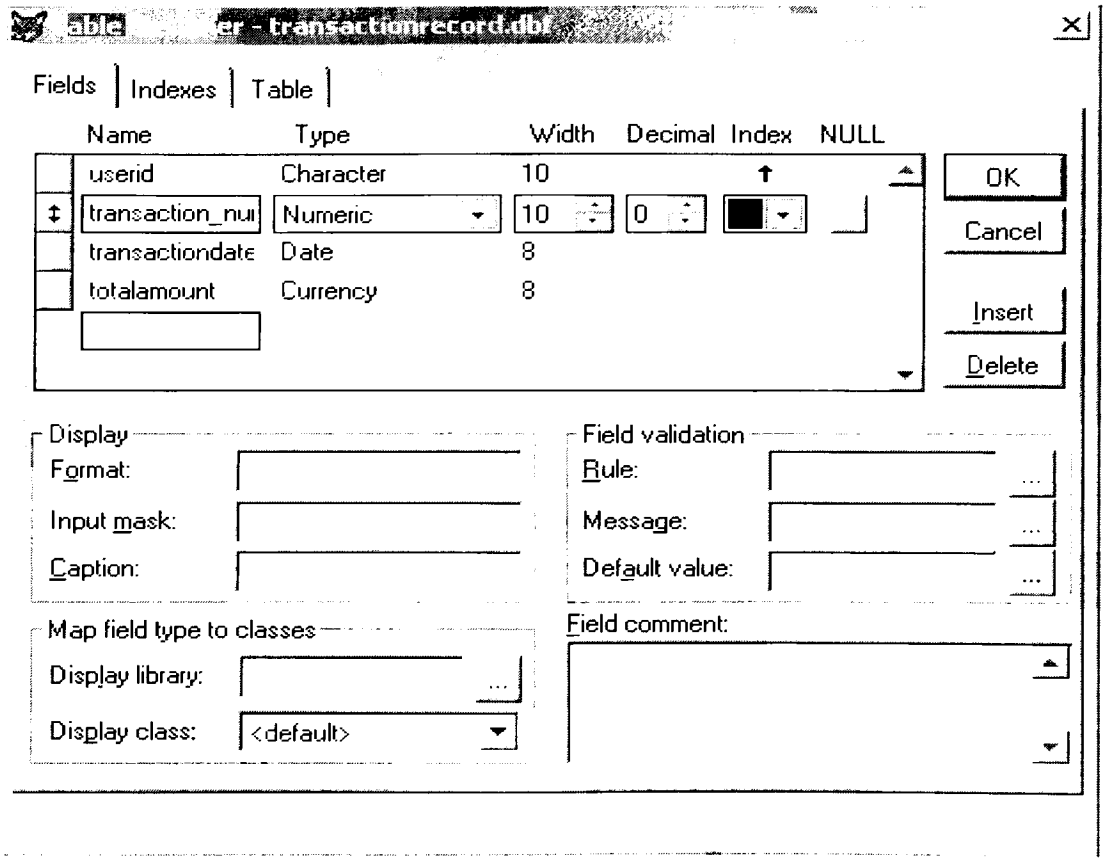

Figure 6.16, Structure of the transactionrecord table.

Itemsstorage table:

The itemsstorage table is used to keep record of each individual item purchased during a particular user transaction. The field transaction_num, which is a primary key in the table transactionrecord, is used as a foreign key in this table. The field transaction_num is used in keeping tract of the buyer and transaction number for each item purchased. The field itemnum, which identifies each individual item, is used as the primary key in this table. The structure for the items table is shown in figure 6.17 below.

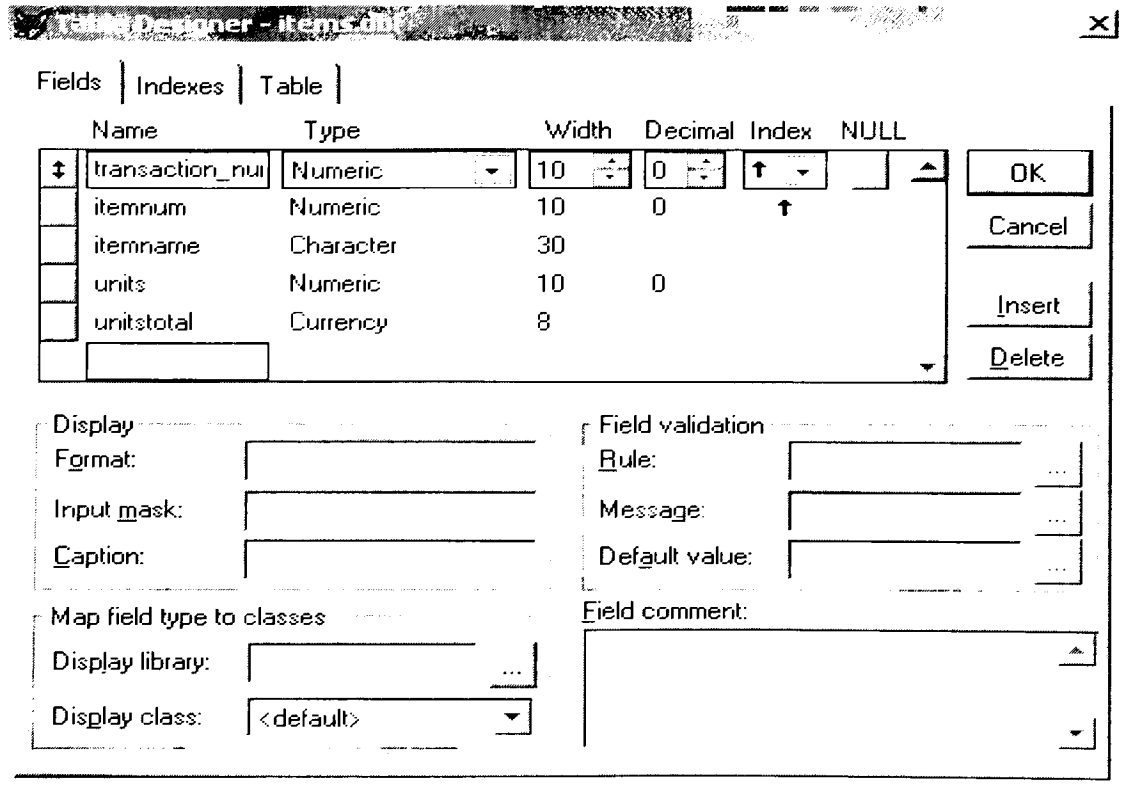

Figure 6.17, Structure of the itemsstorage table

References

Bos, Bert. "XML in Ten Points" <www.w3.org/xml/1999/xml-in-10-points> (March 09 2000).

Cartwright, Adam S. "Server-Side XML in ASP". < http://www.15seconds.com/issue/990527.htm>

Ducharme, Bob. "HTML and XSLT". <http://www.xml.com/pub/a/2000/08/30/xsltandhtml/index.html >(August 30, 2000).

GinsBourg, Niall. "Creating an In-Memory Database Using XML and XPath -- Part 1" < http://www.15seconds.com/Issue/010409.htm>.

Homer, Alex, et al. Professional Active Server Pages 3.0. Birminghan: WROX Press, 1999.

Kropog, Bill. Professional ASP XML. Birminghan: WROX Press, 2000.

Lam, John. "Visual Interdev and ASP" **<http://www.zdnet.com/pcmag/pctech/content/16/22/pp1622.001.html>**

Matt J, Crouch. Web Programming with Asp and Com. New York: Addison Wesley, 1999.

Miller, Ken. Inside Microsoft Visual Interdev. Washington: Microsoft Press, 1998.

Mohr, Stephen. Designing Distributed Applications With XML, ASP, IE5, LDAP and MSMQ. Birminghan: WROX Press, 1999.

Vink, Ian. "Distributing Server Load to the Client with XML and XSL" **<http://www.15seconds.com/issue/010305.htm>**.

"Welcome to XML School" XML Tutorial. **<http://www.w3schools.com/xml/default.asp>**.

Walsh, Norman. "A Technical Introduction to XML" **<http://www.xml.com/pub/a/98/10/guide0.html>**.

6. Conclusion

The nineties was not only a revolution in Internet computing but it also resulted in different number of Operating Systems, platforms and object oriented programming languages. Internet technologies as the World Wide Web have proven to be the biggest Information Systems Architecture today. Web Technologies that make use of Internet protocols such as FTP, HTTP are not only limited to transferring or viewing the contents using a Web browser. Internet can be used as a glue to integrate different operating systems and languages that enterprises may be using. Internet is a run time that is available on any platform and operating system. Internet can also be afforded by enterprises of different sizes. Internet is stateless in nature and does not require a pre-defined connection when sharing information across enterprise boundaries.

In this thesis we have made a study upon Information Sharing Architecture using latest trend in the current Internet technology as XML (eXtensible Markup Language) and SOAP (Simple Object Access Protocol). XML enables us to develop application dependent vocabulary. The biggest power of XML is where the users have the flexibility of defining their own sets of tags. XML is a technology that enables us to store and exchange structured information using Internet protocols such as Hypertext Transport Protocol.

SOAP is an application protocol that is built upon the openness of XML where users have the flexibility of defining their own sets of tags. The SOAP specification defines a set of XML tags that can be used to encode and send message packet using the World Wide Web's HTTP protocol. The SOAP message packet consists of an outer layer, which is called the Envelope. The Envelope consists of an optional Header Element and a mandatory Body Element. The Body Element inside a SOAP message packet is used to hold request, response or error information. The SOAP specification makes use of XML schemas in defining its data types. This enables the invocation of methods and function residing at remote points in the network.

The AsiStore Shopping was implemented as a part of this thesis work to test the feasibility of using XML and SOAP as an Application communication protocols in Enterprise Integration. AsiStore has business partnership with two manufacturers Store1 and Store2. The business partners maintain their products inventory in relational database tables independent of each other. AsiStore uses SOAP messaging to retrieve products update from the business partners. Even though the business partners maintain their database tables independent of each other they have to agree upon some predefined vocabulary. In our AsiStore Shopping application the business partners have agreed to use the AsiStore Products exchange template as the common vocabulary.

This is where the heart of Business-to-Business Integration is headed. XML can be used to define sets of vocabulary that can be used by businesses. Different sets of vocabularies can be defined depending upon the needs and nature of the enterprises. For example, printing industry can define their own set of vocabulary whereas manufacturing

industries can define their own. This idea of defining a standard set of vocabulary enables business to extend across international boundaries. For example, using Standard vocabulary a business in China can interact with a business in Australia easily. XML is affordable by businesses of any size in sharing information with each other. The technologies used in the past in sharing information electronically as EDI (Electronic Document Interchange) could be afforded only by large organizations.

XML not only enables standard vocabulary to be defined for exchanging structured information electronically. XML can be used as an application communication protocol. We have used SOAP which is a protocol based upon XML in our AsiStore Application to retrieve live product information from the business partners. However, just defining a standard set of vocabulary and using a XML based protocol SOAP is not sufficient for businesses to define and discover each other dynamically. Web Services enable businesses to define and discover each other without any prior knowledge of each other. Web Services are a combination of Industry specific business directory UDDI (universal Description, discovery and integration), XML based communication standard SOAP, and a language to define itself, WSDL (Web Services Definition Languages). SOAP can be considered as a communication protocol as well as an implementation when used with Web Services. SOAP is a communication protocol, which carries message content from one end to the other. SOAP is also an implementation that guarantees that he message is delivered and routed successfully.

This thesis work does not go deep into the notion of Web Services that enables businesses to define and discover each other dynamically. This is because Web Services are still work in progress and have a long way to go before they are finally adopted. However, we have been able to study and test the feasibility of SOAP and XML that are used by Web Services extensively. Using XML we were able to implement the AsiStore shopping cart application successfully. We were also able to use SOAP as a messaging protocol to exchange information among business partners successfully.

There is a long way to go before XML and SOAP can be adopted successfully in integrating businesses and Enterprises successfully. These are immature technology and have not yet been tested on Enterprise Level Applications. Both XML and SOAP specifications have changed frequently over the past few years. It will still take a long time before XML and SOAP are adopted successfully in Enterprise level applications. However, as a part of this thesis work we were able to use and implement SOAP successfully even in its premature form.

Internet is platform independent, language neutral and stateless in nature. Most of the Internet standard and specification are not based upon a particular vendor or technology. Using Internet as the Information sharing Architecture promises to solve many of the problems we have faced in the past. Internet is affordable and can be adopted easily by organizations of any sizes. Especially, the birth of Internet Technology as XML and SOAP opens up more room into the future. SOAP is a new technology and still requires a lot of contribution from developers, organizations and institutions. SOAP is a technology that is becoming accepted slowly with a lot of thought and caution. At the

time of writing SOAP is still a work in progress but does seem to be a promising technology especially when it comes to integrating businesses together.

7. Glossary of terms Used

Glossary Of Terms

A

Active Server Pages (ASP)
Microsoft's Server Side Scripting Language to create dynamic websites. An alternative to ASP is JSP.

B

Biztalk
Biztalk is an industry initiative headed by Microsoft to promote Extensible Markup Language (XML) as the common data exchange language for e-commerce and application integration on the Internet (http://whatis.com).

B2C
B2C is short for business-to-consumer, or the retailing part of e-commerce on the Internet. It is often contrasted to B2B or business-to-business. (http://whatis.com)

B2B
B2B stands for business-to-business and is the interaction among businesses in the electronic commerce world.

C

Class
In object-oriented programming, a class is a template definition of the method and variable in a particular kind of object (http://whatis.com).

CDR (Common Data Representation)
CDR stands for Common Data Representation and is used by distributed Object Technologies as CORBA in representing data across the network.

CORBA
Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network (http://whatis.com).

D

DCOM
Distributed Component Object Model (DCOM) is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network (http://whatis.com).

DNS

Domain Name Service Protocol.

DTD

A document type definition (DTD) is a specific definition that follows the rules of the Standard Generalized Markup Language and the extensible Markup Language.

E

EbXML

EbXML is a set of specifications, which conform to a modular electronic business framework.

EDIFACT

Electronic Interchange for Administration, Commerce, and transport.

Electronic Document Interchange

Electronic Document Interchange is the standard for interchanging business data.

Encapsulation

Encapsulation is an object-oriented concept, which involves hiding the implementation details of the objects or class included.

Enterprise Resource Planning (ERP)

ERP is an industry term for the broad set of activities supported by multi-module application software that helps a manufacturer or other business manage the important parts of its business, including product planning, parts purchasing, maintaining inventories, interacting with suppliers, providing customer service, and tracking orders (http://whatis.com).

F

G

H

HTML

HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page (http://whatis.com).

I

Inheritance

Inheritance is an object-oriented concept and which is a way of deriving a child from a parent object or class.

Instantiate

To instantiate is to create such an instance by, for example, defining one particular variation of object within a class, giving it a name, and locating it in some physical place (http://whatis.com).

Interface

An Interface is a set of statements, functions, options, and ways of expressing program instructions and data provided by a program or language for a programmer to use

Internet Information Server

Internet Information Server is a set of Internet Services containing WWW server, FTP Server and Message Queue Server, which runs under the windows platform.

J

Java Server Pages (JSP)

Java Server Pages is a server side-scripting platform available under the Java environment. An alternative to JSP is ASP.

K

L

Light-weight protocol

A simple protocol involving less overhead and easy to understand by all the endpoints and the processes involved in communication.

M

Markup

Markup is a separate activity that takes place after writing and before typesetting.

Marshalling

In computer programming, marshalling is the process of gathering data from one or more application or non-contiguous sources in computer storage, putting the data pieces into a message buffer, and organizing or converting the data into a format that is prescribed for a particular receiver or programming interface (http://whatis.com).

Method

A method is a programmed procedure that is defined as part of a class and included in any object of that class (http://whatis.com).

Metadata

Metadata is definition or description of data.

Metalanguage

Metalanguage is a language used to describe another language.

N

NDR (Network Data Representation)

NDR stand for Network Data Representation and is used by distributed Object Technologies like DCOM in representing data across the network.

NASSL

IBM's Network Accessible Service Specification Language (NASSL) is an IDL to describe Services in the network and complies with the SOA architecture.

O

OASIS

Organization for the Advancement of Structured Information Standards.

Object-Oriented Programming

Object-oriented programming (OOP) is a programming concept based upon organized "objects" instead of "actions," and data rather than logic.

Objects

In object-oriented programming (OOP), objects are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process (http://whatis.com).

P

Parser

In computer technology, a parser is a program, usually part of a compiler, that receives input in the form of sequential source program instructions, interactive online commands, markup tags, or some other defined interface and breaks them up into parts (for example, the nouns (objects), verbs (methods), and their attributes or options) that can then be managed by other programming (for example, other components in a compiler) (http://whatis.com).

Payload

Payload is a packet or unit of data sent across the Internet.

Q

R

RMI

RMI (Remote Method Invocation) is a way that a programmer, using the Java programming language and development environment, can write object-oriented programming in which object on different computers can interact in a distributed network (http://whatis.com).

S

Schemas

An alternative to Document Type definition, which can hold data types as in any object-oriented language.

Service-Oriented Architecture

Service-Oriented Architecture is a conceptual model for implementing electronic business dynamically.

Serialization

The process of transforming an object into a stream of bytes is called serialization (http://java.about.com/compute/java/library/weekly/aa101700a.htm).

Simple Object Access Protocol (SOAP)

SOAP is a specification used in communicating between different platform and languages over the World Wide Web. SOAP uses HTTP and XML to transport message between endpoints.

SGML

Standard Generalized Markup Language (SGML) is a standard for how to describe a set of tags or markup.

T

U

Unicode

Unicode is the international standard for the interchange, display, and processing of written texts in the different languages of the modern world. The current Unicode standard contains 34,168 distinct coded characters, which are derived from 24, supported language scripts.

UN/CEFACT

United Nations body for Trade Facilitation and Electronic Business.

UDDI

Universal Description, Discovery, and Integration (UDDI) is an XML-based registry for businesses worldwide to list themselves on the Internet (http://whatis.com).

V

W

W3C

World Wide Web Consortium.

WDS

Stands for Well Defined Services and are XML documents used by services to publish data to the service brokers.

WSDL

Stands for Web Services Definition Language and is an xml-based set of specification to describe and define Web Services.

X

X12

A standard initiated by the American National Standard Institute to facilitate the electronic exchange of business information.

XML

Stands for Extensible Markup language and is a flexible way to create information units that share both format and data on the World Wide Web.

XML Protocol Activity

The goal of XML Protocol is to develop technologies which allow two or more peers to communicate in a distributed environment, using XML as it encapsulation language (http://www.w3.org/2000/xp/).

XSLT

XSLT stands for Extensible Style Sheet Transformation and used in transforming xml documents into another format.

8. Appendix

Filename: Functionincludes.asp

```
<%

'*****************************************************************************************
' Function:showlistofitems (ByRef xmldom)
' Author:  Puskar Adhikari Purpose:Dispalys the availabe items that can be
' shopped. This is done after  receiving the item listing from all the servers
after sending and  receiving the SOAP messages
'Parameters: Return Values: will return html table as a string which can be
'displayed on the browser
'Preconditions:
' Postconditions:
' Side Effects:
' Modified: January 31st, 20001
'*****************************************************************************************

Function showlistofitems(ByRef xmldom)
   dim counter
   dim sbg 'background colour
   dim productcode
   dim priceofproduct
   dim nameofproduct
   dim myquerystring

   'variable to set the row color
   greyrow = "#efefef"
   whiterow = "#ffffff"

   myquerystring = "" 'initialize query string
   counter = 0   'initialize counter to 0
   stableheader =  ""
         stableheader = stableheader & "<table align=center" & QUOT&"
         cellpadding" &_
         QUOT & "=4" & QUOT & " width" & QUOT & "=600"& QUOT
         & " bgcolor=" & QUOT & "#0066FF" & QUOT & ">"
   stable = stable & "<tr>"
   Set root = xmldom.documentElement 'set the root element object
   Set firstElement = root.Childnodes.item(0) 'set the first child element object

   'this section of the code will put the appropriate table header to be
   'displayed

   for each child in firstElement.childNodes

         if child.nodename<> "units" then
        header = ""
        'this will just put the header for the table of items
        if child.nodename="productid"  then
            header = "Item ID"
        elseif child.nodename="productname" then
           header =  "Product Name"
        else
           header = "Product Price"
        end if


         stable = stable & "<td valign=" & QUOT & "top" & QUOT & "
         bgcolor=" & QUOT & "#0066FF" & ">"
```

```
                'this will write down the attribute header
                stable = stable & "<font color = " & QUOT & "#ffffff><B>"
                stable = stable & header
                stable = stable & "</font></B></TD>"

                end if

next
  stable = stable & "<td valign=" & QUOT & "top" & QUOT & " bgcolor=" & QUOT
      & "#0066FF" & ">"
                'this will write down the attribute header
                stable = stable & "<font color = " & QUOT & "#ffffff><B>"
                stable= stable & "Add Item"
                stable = stable & "</font></B></TD>"

stable = stable & "</tr>"

for i=0 to (root.childnodes.length-1)

   'this if else statemnet will alternate color of the rows
                'this wil give us different color for each row
   if (counter mod 2) = 0 then
                sbg = greyrow
      else
                sbg = whiterow
      end if

      'sbg = "#efefef"  'background color
   'set each individual node

                set mynodeelements  = root.Childnodes.item(i)

                stable = stable & "<tr bgcolor = " & QUOT & sbg & ">"

                for each child in mynodeelements.childNodes
                if child.nodename <> "units" then   'dont show the num of units
                        stable = stable & "<td STYLE='font-family:Arial, sans-serif;
                         font-size: 10pt' valign =" & QUOT & "center>
                        if child.Nodename = "productid" then
                        'this is to identify the component in the form
                        productcode = child.text
                        end if

                if child.Nodename = "price" then
                stable= stable & Formatcurrency(child.text)
                priceofproduct = child.text
                'elseif child.Nodename = "ProductID" then
                'stable= stable & child.text
                elseif child.Nodename = "productname" then
                stable= stable & child.text
                nameofproduct = child.text
                else
                stable= stable & child.text
                end if

                'this will store the product code for the
                'value for butttons and ID
                stable = stable & "</td>"
```

```
                    end if

                    stable = stable & "</font>"

                    next

                    stable = stable & "<td valign =" & QUOT & "middle>" & QUOT

                    'building the query string for the item selected
                    myquerystring = "<a href = 'cart.asp?code="&productcode
                    myquerystring = myquerystring & "&name="&nameofproduct
                    myquerystring = myquerystring & "&price="&priceofproduct&"'>"

                    stable = stable & myquerystring

                    stable = stable & "<img border=" & QUOT & "0" & QUOT
                    stable = stable & " src=" & QUOT & "../images/forwardbutton.gif>"
                    stable = stable & "</img>"
                    stable = stable & "</a>"

                    stable = stable & "</td>"


                    set mynodeelements = nothing

                    stable = stable & "</tr>"
                    counter = counter + 1
    next

        stableheader = stableheader & stable & "</TABLE>"


        stableheader = stableheader & "<BR>"

        stableheader = stableheader & "<table align=center width = 600>"
        stableheader = stableheader & "<TR><TD align=center>"
         stableheader = stableheader & "<a href = 'cart.asp" & "'>"
        stableheader = stableheader & "<img border=" & QUOT & "0" & QUOT
        stableheader = stableheader & " src=" & QUOT & "../images/viewcart.gif>"
        stableheader = stableheader & "</img>"
        stableheader = stableheader & "</a>"
        stableheader = stableheader & "</TR></TD>"
        stableheader = stableheader & "</table>"

        showlistofitems=stableheader

End Function


'*********************************************************************************************
' Function: showshoppingbasket (ByRef xmldom)
' Author:  Puskar Adhikari
' Purpose: dispalys the shopping cart
' Parameters:    DOM object passed by reference
' Return Values: Will return the shopping basket HTML code as a string
' which can be displayed on the browser
' Preconditions:
' Postconditions:
```

```
' Side Effects:    none
' Modified: January 31st, 20001
'*****************************************************************************************

Function showshoppingbasket(ByRef xmldom)

   dim counter
   dim sbg 'background colour
   dim root
   dim productcode

   dim runningTotal

   dim greyrow   'color of the alternating row
   dim whiterow

   checkifempty = ""
   runningTotal = 0
   counter = 0
   greyrow = "#efefef"
   whiterow = "#ffffff"

   puttextvalue=""
   counter = 0   'initialize counter to 0
   stableheader =  ""
         stableheader = stableheader & "<table align=center " & QUOT&" cellpadding" &_
         QUOT & "=4" & QUOT & " width" & QUOT & "=600"& QUOT & " bgcolor="
          & QUOT & "#0066FF" & QUOT & ">"
   stable = stable & "<tr>"
   Set root = xmldom.documentElement 'set the root element object
   Set firstElement = root.Childnodes.item(0) 'set the first child element object
   'Now this will putin the table header

   for each child in firstElement.childNodes
      stable = stable & "<td valign=" & QUOT & "top" & QUOT & " bgcolor="
                  & QUOT & "#0066FF" & ">"
                  'write down the attribute header
                  stable = stable & "<font color = " & QUOT & "#ffffff><B>"
                  stable= stable & child.nodename
                  stable = stable & "</font></B></TD>"
   next
                  stable = stable & "<td valign=" & QUOT & "top" & QUOT & " bgcolor=" &
                  QUOT & "#0066FF" & ">"
                  'write down the attribute header
                  stable = stable & "<font color = " & QUOT & "#ffffff><B>"
                  stable= stable & "Unit Total"
                  stable = stable & "</font></B></TD>"

                  stable = stable & "<td valign=" & QUOT & "top" & QUOT & "
                   bgcolor=" & QUOT
                   & "#0066FF" & ">"
                  'write down the attribute header
                  stable = stable & "<font color = " & QUOT & "#ffffff><B>"
                  stable= stable & "Remove"
                  stable = stable & "</font></B></TD>"

   stable = stable & "</tr>"
```

```
if root.childnodes.length = 0 then

        checkifempty = true
end if


for i=0 to (root.childnodes.length-1)
        'this if else statemnet will alternate color of the rows
    if (counter mod 2) = 0 then
        sbg = greyrow   'background color
    else
        sbg = whiterow
    end if

     'set each individual node

                set mynodeelements  = root.Childnodes.item(i)

                stable = stable & "<tr bgcolor = " & QUOT & sbg & ">"

                for each child in mynodeelements.childNodes

                stable = stable & "<td STYLE='font-family:Arial, sans-serif; font-size: 10pt'
                valign =" & QUOT & "top>"
                        if child.Nodename = "ProductID" then
                                'this is to identify the component in the form
                                productcode = child.text
                        end if

                        if child.Nodename = "Units" then
                        puttextvalue = child.text
                        'put text boxes and update button
                        stable = stable  & "<input TYPE= " & QUOT & "TEXT" &_
                        " NAME=" & QUOT & "box" & productcode& QUOT & "
                        size= " & QUOT & " 4 " & QUOT & " value= " & QUOT &_
                        puttextvalue & QUOT & ">"
                        stable = stable &<BR>"
                        stable = stable & "<input TYPE=" & QUOT & "IMAGE" &_
                        " NAME=" & QUOT & " update" & productcode & QUOT & "
                         value= " & QUOT &_
                        productcode & QUOT & " SRC= " & QUOT  &
                         "../images/save.gif" & QUOT & ">"
                        elseif child.Nodename = "Price" then
                        myunitprice = child.text    'unit price for each item
                        stable= stable & Formatcurrency(child.text)
                        else
                        stable= stable & child.text
                        end if
                        'this will store the product code for the
                        'value for butttons and ID
                        stable = stable & "</td>"
                next

                stable = stable & "<td valign =" & QUOT & "top>"

                'stable = stable  & "<input TYPE= " & QUOT & "TEXT" &_
                "' NAME=" & QUOT & "pricebox" & productcode& QUOT & " size= "
                & QUOT & "10" & QUOT & " value= " & QUOT &_
```

```
                'formatcurrency(puttextvalue*myunitprice) & QUOT & ">"
                stable = stable & formatcurrency(puttextvalue*myunitprice)
                'sum Total
                runningTotal = runningTotal + (puttextvalue*myunitprice)

                stable = stable & "</td>"


                stable = stable & "<td valign =" & QUOT & "top>"
                stable = stable & "<input TYPE= " & QUOT & " IMAGE" &_
                " NAME=" & QUOT & " delete" & productcode & QUOT & "
                value= " & QUOT &_
                productcode & QUOT & " SRC= " & QUOT
                  &  "../images/remove.gif" & QUOT & ">"
                stable = stable & "</td>"

                set mynodeelements = nothing

                stable = stable & "</tr>"
                counter = counter + 1
next

        stableheader = stableheader & stable & "</TABLE>"

        '2nd table
        stableheader = stableheader & "<table align=center width = 600>"
        stableheader = stableheader & "<TR><TD>"
stableheader = stableheader & "<Font color=blue>"
        stableheader = stableheader & "<P align=right>"
        stableheader = stableheader & "<B>Total Order:</B>"
        stableheader = stableheader & formatcurrency(runningTotal)
        stableheader = stableheader & "</p>"
        stableheader = stableheader & "</Font>"
        stableheader = stableheader & "</TR></TD>"
        stableheader = stableheader & "</table>"


        '3rd link table


        stableheader = stableheader & "<table align=center width = 600>"

        stableheader = stableheader & "<TR><TD align=center>"
stableheader = stableheader & "<a href = 'showitems.asp" & "'>"
        stableheader = stableheader & "<img border=" & QUOT & "0" & QUOT
        stableheader = stableheader & " src=" & QUOT & "../images/viewitems.gif>"
        stableheader = stableheader & "</img>"
        stableheader = stableheader & "</a>"

        'stableheader = stableheader & "<a href = 'readuserinfo.asp" & "'>"
        'stableheader = stableheader & "<img border=" & QUOT & "0" & QUOT
        'stableheader = stableheader & " src=" & QUOT & "../images/putorder.gif>"
        'stableheader = stableheader & "</img>"
        'stableheader = stableheader & "</a>"

       stableheader = stableheader & "</TR></TD>"
```

```
        stableheader = stableheader & "<TR><TD align=center>"
        stableheader = stableheader & "<a href = 'showsummarytouser.asp" & "'>"
        stableheader = stableheader & "<img border=" & QUOT & "0" & QUOT
        stableheader = stableheader & " src=" & QUOT & "../images/putorder.gif>"
        stableheader = stableheader & "</img>"
        stableheader = stableheader & "</a>"

        stableheader = stableheader & "</TR></TD>"
        stableheader = stableheader & "</table>"



        if checkifempty=true then
                showshoppingbasket="empty"
        else
                showshoppingbasket=stableheader
        end if

End Function


'*********************************************************************************************
' Function: communicatewithServer (ipaddressofmachine)
' Author:  Puskar Adhikari
' Purpose: This function will ask for  the IP adress of the remote machine and then
' it will communicate with the machine residing at the particular IP address uisng SOAP
' as the messaging protocol.  This used the htpp get/post method in communication
' Parameters: ipaddressof othe machine to communicate with
' Return Values: will return xml string containing the data
' Preconditions: none
' Postconditions: none
' Side Effects:    none
' Modified:February 07, 20001
'*********************************************************************************************

Function communicatewithServer(ipaddress)

        dim soapmessageString
        const ip = "http://129.21.92.46/productionservers/giveplainxml.asp"
        soapmessageString=""
        communicatewithServer = ""
        'will return a string
        'initialize the xml ojects
        Set xmlhttp = Server.CreateObject("Microsoft.XMLHTTP")
        set xmldom = Server.CreateObject("MSXML2.DOMDocument")

        'set up the soap message header
  xmlhttp.open "POST",ipaddress, false
        xmlhttp.setRequestHeader "Man", POST & " " & ipaddress & "HTTP/1.1"
        xmlhttp.setRequestHeader "SOAPMethodName", "returnxmlstring"
        xmlhttp.setRequestHeader "MessageType", "CALL"
        xmlhttp.setRequestHeader "ContentType", "text/xml"

  'build the soap Envelope and send the message to the server

  soapmessageString = buildSoapEnvelope()
        xmlhttp.send(soapmessageString)
```

```
        if xmlhttp.status = 200 then
                Set xmldom = xmlhttp.responseXML
                'Response.Write xmldom.xml
                'soapmessageString = xmlhttp.responseXML.xml
                communicatewithServer = fitintoasiproductformat(xmldom)

                'Set soapmessageString = xmldom.documentElement
                'communicatewithServer =  soapmessageString.xml
                'Response.Write soapmessageString.xml
                'display the resulting string in HTML format
        else
                communicatewithServer = "FAULTELEMENT"
        End if


        Set xmlhttp = nothing
        Set xmldom = nothing


End function


'*********************************************************************************************
' Function: buildSoapEnvelope()
' Author: Puskar Adhikari
' Purpose: This function builds the required soap Message Package
' Parameters: none
' Return Values: string containing the soap message
' Preconditions: none
' Postconditions: none
' Side Effects:
' Modified: February 07, 20001
'*********************************************************************************************

Function buildSoapEnvelope()

  buildSoapEnvelope = buildSoapEnvelope &
   "<SOAP:Envelope xmlns:SOAP=""http://schemas.xmlsoap.org/soap/envelope/""
   xmlns:m=""http://lathe.cims.rit.edu/example"">"
   buildSoapEnvelope = buildSoapEnvelope & "<SOAP:Body>"
   buildSoapEnvelope = buildSoapEnvelope & "<m:returnxmlstring>"
   buildSoapEnvelope = buildSoapEnvelope & "</m:returnxmlstring>"
   buildSoapEnvelope = buildSoapEnvelope & "</SOAP:Body>"
   buildSoapEnvelope = buildSoapEnvelope & "</SOAP:Envelope>"


End Function


'*********************************************************************************************
' Function: fitintoasiproductformat()
' Author:  Puskar Adhikari
' Purpose: This function builds the required soap Message Package
' This manages the string in the format <Product><Productid>1001</ProductID>
'....</Product>
' This is used extract the datas from the soap message will return a pure
```

```
'string which
' can be loaded into our xml object
' Parameters: none
' Return Values: XML string that contains the string of product informations
' Preconditions: none
' Postconditions: none
' Side Effects:
' Modified: February 07, 20001
'*********************************************************************************************

Function fitintoasiproductformat(Byref xmldomobject)

 Set soapmessageString = xmldomobject.documentElement.childNodes.
item(0).childNodes.item(0)
 myonlystring = ""
 'now we will filter our information from the SOAP message . We will get rid of
 'the envelope and the method name.  It will also chek to see if it has
 'any fault element associated with it

 for i=0 to (soapmessageString.childnodes.length-1)

        myonlystring = myonlystring & "<Product>"
        set mynodeelements  = soapmessageString.childnodes.item(i)
        for each child in mynodeelements.childnodes
                myonlystring = myonlystring & "<" & child.nodename & ">"
                 myonlystring = myonlystring & child.text
                myonlystring = myonlystring & "</" & child.nodename & ">"
        next
          myonlystring = myonlystring & "</Product>"

        next

   fitintoasiproductformat =  myonlystring

End Function



'*********************************************************************************************
' Function: pingMachine (ByRef xmldom)
' Author:  Puskar Adhikari
' Purpose: Pings machine to see if it is responding
' it pings the machine 5 times and returns 0 if the machine is functional
' Parameters: IP address of the host is passed
' Return Values: 0 if machine is responding and 1 if it is not
' Preconditions: Nothing
' Postconditions: nothing
' Side Effects: none
' Modified: February 25, 2001
'*********************************************************************************************

Function pingMachine(machineip)


        Set WShShell = Server.CreateObject("WScript.Shell")

        IP = machineip ' or whatever you want to ping
        RetCode = WShShell.Run("C:\Inetpub\cgi-bin\DoPing.bat "
```

```
        & IP & " " & FileName, 1, True)

        if RetCode = 0 then
                pingMachine = RetCode
        else
                pingMachine = 1
        end if


End Function

'*************************************************************************
' Function: checkifloginused()
' Purpose: checks to see if the username has already been taken by
' someone before
' Parameters: none
' Return Values: none
' Preconditions: none
' Postconditions: sets a session variable with error message if the
' username and
' password is already in use
' Side Effects: none
' Modified: April 15, 2001
'*************************************************************************

Function validateNewpassword(txtpassword1, txtpassword2)

        dim error_string
        error_string = "valid" 'password is valid by default


   If Len(txtpassword1) < 6 Then

        error_string = "<BR>Your password should be atleast 6 characters in length."

        elseif Len(txtpassword1) >= 6 then

        'if bothepasswords are same
        if StrComp(txtpassword1, txtpassword2, 1) = 0 then
                'validate password using regular expression now

                Set regEx = New RegExp
                ' Create a regular expression.
                regEx.Pattern = "[a-zA-Z0-9\?\!\$@]+"
                Set Matches = regEx.Execute(strValue)  ' Execute search.
                For Each Match in Matches                ' Iterate Matches collection.
                        If Match.Value <> strValue Then
                        error_string = error_string
                        & "<BR>Your password can only contain letters, numbers and the
                         following symbols: ?, !, $, @.  Your password cannot contain spaces."
                        End If
                Next

                else
                        'invalid password
                        error_string = "<BR>Your passwords do notmatch. Please Re-enter!!"
```

```
                end if

        end if

        validateNewpassword = error_string

End Function

'*****************************************************************************************************
' Function: checkifloginused()
' Purpose: checks to see if the username has already been taken by someone before

' Parameters: none
' Return Values: none
' Preconditions: none
' Postconditions: sets a session variable with error message if the username and
' password is already in use
' Side Effects: none
' Modified: April 15, 2001
'*****************************************************************************************************

Function checkifloginused(txtusername)

        Dim Conn, cmd, RS

        checkifloginused = false     'login is not used

        Set Conn = Server.CreateObject("ADODB.Connection")
        Conn.Open Application("myConnection_ConnectionString")

        SQL = "SELECT * FROM usertable WHERE username="&""&txtusername&""

        Set RS = Conn.Execute(SQL)

        'RS.movefirst

        do until RS.eof
    'if username has already been used
                If trim(RS("username"))=txtusername then
                        checkifloginused = true
                end if
                Rs.movenext

        loop

        Conn.Close   'close database connection
        Set Conn = Nothing


End Function

'*****************************************************************************************
' Function: addNewmember
' Author:  Puskar Adhikari
' Purpose: This machine does the user validation
' Parameters: IP address of the host is passed
' Return Values: flase if invalid and true if valid
' Preconditions: false
```

```
' Postconditions: depends
' Side Effects: none
' Modified: April 15, 2001
'*****************************************************************************************

Function addNewmember(txtusername, txtpassword)

        Dim Conn, cmd, RS
        addNewmember = false
        if checkifloginused(txtusername) = false then

                Set Conn = Server.CreateObject("ADODB.Connection")
                Conn.Open Application("myConnection_ConnectionString")
                'build the SQL ststement
                SQL = "INSERT INTO usertable (username, password) "
                SQL = SQL & "VALUES (" & """ & txtusername & """
                SQL = SQL & "," & txtpassword & ")"
                'add the username and password to database
                Set RS = Conn.Execute(SQL)
                Conn.Close   'close database connection
                Set Conn = Nothing
                Set RS = Nothing
                addNewmember = true

        end if


End Function



'*****************************************************************************************
' Function:        isLegaluser (username, password)
' Author:  Puskar Adhikari
' Purpose:         This machine does the user validation
' Parameters: IP address of the host is passed
' Return Values: flase if invalid and true if valid
' Preconditions: false
' Postconditions: depends
' Side Effects: none
' Modified:        February 25, 2001
'*****************************************************************************************


Function isLegaluser(txtusername, txtpassword)

   Dim Conn, cmd, RS
          isLegaluser = false  'returns false by default
   Set Conn = Server.CreateObject("ADODB.Connection")
   Conn.Open Application("myConnection_ConnectionString")

  SQL = "SELECT * FROM usertable WHERE trim(username)="'
  & txtusername & "' AND trim(password) ="' & txtpassword & """

  'execute SQL statement
   Set RS = Conn.Execute(SQL)
```

```
    If RS.EOF Then
    'if invalid reset the session variables
            isLegaluser=False
    Else
            isLegaluser=True
    End If

            RS.Close
            Set RS = Nothing
            Conn.Close   'close database connection
    Set Conn = Nothing

End Function


'*****************************************************************************************
' Function:        hasaccessRight()
' Author:  Puskar Adhikari
' Purpose:         This machine checks to see if the user is officially logged on and
' has a session. For security measures
' Parameters: none
' Return Values: flase if invalid and true if valid
' Preconditions: false
' Postconditions: depends
' Side Effects: none
' Modified:        February 25, 2001
'*****************************************************************************************

Function hasaccessRight()

            hasaccessRight = false 'false by default

            if Session("isvaliduser") = true then

                        hasaccessRight = true

            else

                        hasaccessRight = false

            end if

End Function



'*************************************************************************
' Function:        isSelected(strString1,strString2)
' Purpose:         This function checks to see if the two strings are equal.
' Parameters: strString1, strString2
' Return Values: returns string "SELECTED" if the two strings are equal.
' Preconditions: none
' Postconditions: none
' Side Effects:    none
' Modified:        February 27, 2001
'*************************************************************************

Function isSelected(strString1,strString2)
```

```
        if StrComp(strString1, strString2, 1) = 0 then
                isSelected = "SELECTED"
        end if

End Function

'*************************************************************************
' Function:       isBlankmessage(field)
' Purpose:        This function makes sure that filled is not left empty
' if empty it will bail error message
' Parameters: filed which is empty
' Return Values: returns the errorcode
' Preconditions: none
' Postconditions: none
' Side Effects:   none
' Modified:       February 27, 2001
'*************************************************************************************************

Function isBlankmessage(field)

  myString = ""
  myString = myString & "<BR>Your " & field & " is required to complete your request"
  isBlankmessage = myString

End Function



'*************************************************************************************************
' Function:       ValidEmail(strvalue)
' Purpose:        This function makes sure that the entered email is in  the right
' format
' if empty it will bail error message
' Parameters: filed which is empty
' Return Values: returns the errorcode
' Preconditions: none
' Postconditions: none
' Side Effects:   none
' Modified:       February 27, 2001
'*************************************************************************************************


Function ValidEmail(strValue)

        validEmail = ""
        if strValue = "" then
                'append to existing error
                'email address optional ok
                validEmail = ValidEmail & ""
        else
                'there is something, see if it's formed right
                'proper form is left@right.com
                'see if there is an at sign
                intSerachPos = InStr(1,strValue,"@",1)
                if intSerachPos = 0 then
                        validEmail = validEmail & "<BR>Your e-mail
                        address did not have an @ sign in it"
```

```
                else
                        if InStr(intSerachPos, strValue,".",1) = 0 then
                                validEmail = validEmail & "<BR>The part of your e-mail
                        address to the right of the @ sign is not properly formed"
                        end if
                end if
        end if

End Function


'*****************************************************************************************************************
' Function: Isnumericform
' Purpose: This checks to see if the user input field is in numeric format
' Parameters: strFieldName, strvalue
' Return Values: none
' Preconditions: none
' Postconditions: returns the errormessage saying numeric field is required
' numeric
' Side Effects: none
' Modified:       February 27, 2001
'*****************************************************************************************************************

Function Isnumericform(strFieldName, strvalue, strvalueentered)

    if  not IsNumeric(strvalueentered) then

        Isnumericform = Isnumericform & "<BR>Your "& strFieldName & " should be numeric"

        else

        if (strvalue="phone1" and Len(strvalueentered)<3) then
                Isnumericform = Isnumericform & "<BR>Your "& strFieldName
                 & " should be 3 digits"
        elseif (strvalue="phone2" and Len(strvalueentered)<3) then
                Isnumericform = Isnumericform & "<BR>Your "& strFieldName
                & " should be 3 digits"
        elseif (strvalue="phone3" and Len(strvalueentered)<4) then
                Isnumericform = Isnumericform & "<BR>Your "& strFieldName
        & " should be 4 digits"
        end if

    end if

End Function


'*****************************************************************************************************************
' Function: validateStringitem(strValue, whattovalidate)
' Author:  Puskar Adhikari
' Purpose: This checks to see if the user input field is right format
' and this uses the regular expression validation
' Parameters: value entered and the kind of item to validate using regular
' expression
' Return Values: none
' Preconditions: none
```

```
' Postconditions: returns the errormessage saying numeric field is required
' numeric
' Side Effects: none
' Modified: February 27, 2001
'*****************************************************************************************************


Function validateStringitem(strValue, whattoValidate)

        validateStringitem = ""

        if whattoValidate = "creditcardnumber" then
                limit = 16
                myExpression = "[0-9]+"
                erroroutputtype = "Credit Card Number"
        elseif whattovalidate = "zipcode" then
                limit = 5
                myExpression = "[0-9]+"
                erroroutputtype = "Zip Code"
        end if

        If Len(strValue) < limit Then
                validateStringitem = "<BR>Your " & erroroutputtype &
                 " should contain " & limit & " digits"
        Else
                Set regEx = New RegExp
        ' Create a regular expression.
                regEx.Pattern = myExpression
                Set Matches = regEx.Execute(strValue)
                ' Execute search.

                For Each Match in Matches
                ' Iterate Matches collection.
                If Match.Value <> strValue Then
                        validateStringitem = validateStringitem & "<BR>Your" &
                         erroroutputtype & " can only conatin numbers"
                End If
                Next

        End If

End Function


'*****************************************************************************************
' Function: sendemail(messagestring, sender, receiver)
' Author:  Puskar Adhikari
' Purpose: This is function which will send email to the user and can be used for
' sending confirmation messages
' Parameters:
' Return Values:
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: January 31st, 20001
'*****************************************************************************************
```

```
Function sendemail(messageString, sender, receiver, subject)

        Dim objCDOMail
        Set objCDOMail = Server.CreateObject("CDONTS.NewMail")

        ' Set the properties of the object
        objCDOMail.From = sender
        objCDOMail.To = receiver
        objCDOMail.Subject = subject
        objCDOMail.Body = messageString


        objCDOMail.Send

        ' Set the object to nothing because it immediately becomes
        ' invalid after calling the Send method.
        Set objCDOMail = Nothing
        sendemail = 1


End Function


%>
```

Filename: Cart.asp

```
<%@ Language=VBScript %>
<!--#include file="functionincludes.asp"-->

<%
Response.Buffer = true
dim id, name, price, unit
emptymessage = ""
'check if the user has logged in or not

if hasaccessRight = false then

        Session.Abandon
        Response.Redirect "memberlogin.asp"

end if

set xmldom = Server.CreateObject("MSXML2.DOMDocument")
Set myxmlstring = Session("myxmlstring")

updateplease = myxmlstring.givexmlforDisplay()  'display xml by default
for i=1000 to 1050

    builddelete = "delete" & CStr(i) & ".X"
    buildupdate = "update" & CStr(i) & ".X"
    buildbox = "box" & CStr(i)

        if (Request(builddelete))  <> "" then
           productcode = i
           updateplease = myxmlString.deleteItem(productcode)
        end if

        if (Request(buildupdate))  <> "" then
            productcode = i
            'Response.Write "<BR>We have " & trim(request(buildbox))
            updateplease = myxmlString.updateItem(productcode,request(buildbox))
        end if

next



if Request("code")<>"" then

        id = Request("code")
        name =  Request("name")
        price =  Request("price")
        unit = 1


        if myxmlString.getItem(id) then   'see if the item is already in the shopping basket
           'get the number of units already in the basket
           mynumunits = myxmlString.returnNumunits(id)

           if mynumunits <> 0 then
           'add one to the existing
                        mynumunits = mynumunits + 1
```

```
                    updateplease = myxmlString.updateItem(id,mynumunits)
        end if

    else
                        'if false then just add it to the list
            updateplease = myxmlstring.addItem(id, name, price, unit)

    end if

end if


xmldom.loadXML(updateplease)
Set checkmybasket = xmldom.documentElement

if checkmybasket.childnodes.length <> 0 then
   'show shopping
        display = showshoppingbasket(xmldom)
else
        'Response.Redirect "cart.asp"
        cartisempty = "empty"
end if

Set xmldom = nothing
%>

<html>
<body bgcolor="ghostwhite">
<CENTER>
<!--#include file="useridbox.asp" -->
</CENTER>

<form name="shop" id="shop" action="">

<%=display%>

<%
if cartisempty = "empty" then
%>

<font face="Arial, Helvetica, sans-serif" size="3">
<font color='#FF0000'>Your Shopping Basket is Empty!!</font>
</font>

<br>

<table align="center" width="600">
<tr>
<td align="center">
<a href="showitems.asp">
<img border="0" src="../images/viewitems.gif" WIDTH="150" HEIGHT="16"></a>
</td>
</tr>
</table>

<%
end if
%>
```

```
</form>

<!--#include file="banners.asp" -->

</body>
</html>
```

Filename: mailer.asp

```
<%@ Language=VBScript %>
<%Response.Buffer = true %>
<!--#include file="functionincludes.asp"-->
<%
  dim error
  dim yourname, youremailaddress, yoursubject
  errormessage = ""

  if request("submit1") <>"" then
        yourname = request("name")
        if yourname ="" then
              errormessage = errormessage & isBlankmessage("Name")
        end if
        youremailaddress = request("email")

        if youremailaddress ="" then
              errormessage = errormessage & isBlankmessage("e-mail address")
        else
              errormessage = errormessage & ValidEmail(youremailaddress)
        end if

        yoursubject = request("subject")
        yourmessage = request("yourmessage")

        if errormessage = "" then
              confirm = sendemail(yourmessage, youremailaddress, "pna2915@rit.edu", "")
              flag = "complete"
        else
              flag = "incomplete"
        end if

  else
        flag = ""
  end if

%>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>
<BODY bgcolor=ghostwhite>


<i>
        <font face="Arial, Helvetica, sans-serif" size="2">
        <font color="#FF0000"><%=errormessage%></font>
        </font>
</i>

<BR>
<BR>
<center>

<% if flag="incomplete" or flag="" then  %>
```

```
<table bgcolor=#ddb7ba border=0 cellpadding=0 cellspacing=0 width=500>

<tr bgcolor=#fff5f6>
<td  align=middle>
<font color=#6f6f6f size=6>
Welcome to AsiStore<b> <%=Session("username")%>
</font></B>
<br>
<font color=#6f6f6f size=-2>
Today is
<%=FormatDateTime(now(),vblongdate)%>

<br>

<font color=green size=-2><%=time%>
</font>
<br>You have been on from <%=Request.ServerVariables("REMOTE_ADDR")%>
</font></FONT>

</td>
</tr>

</table>

<BR>




<FORM action="mailer.asp" id=form1 name=form1 method="post">
<TABLE align=center bgcolor=lavender border=0 cellPadding=3 cellSpacing=0 width=500>
 <TBODY>

 <TR bgcolor=lavender>
  <TD vAlign=top width="37%">
   <P><FONT face=Arial>Your name:</FONT></P></TD>
  <TD vAlign=top width="37%">
   <P>
   <INPUT maxLength=50 value="<%=yourname%>" name="name"> </P></TD>
  <TD rowSpan=4 vAlign=top width="26%">  </TD></TR>
 <TR bgcolor=lavender>
  <TD vAlign=top width="37%">
   <P><FONT face=Arial>E-mail:</FONT></P></TD>
  <TD vAlign=top width="37%">
   <P>
   <INPUT maxLength=50 value="<%=youremailaddress%>" name="email"> </P></TD></TR>
 <TR bgcolor=lavender>
  <TD height=22 vAlign=top width="37%">
   <P><FONT face=Arial>Subject:</FONT></P></TD>
  <TD height=22 vAlign=top width="37%">
  <INPUT value="<%=yoursubject%>"  maxLength=50 name=subject></TD>
  <TD height=22 vAlign=top width="37%">
  <FONT color=#c0c0c0>.</FONT> </TD></TR>
 <TR bgcolor=lavender>
  <TD colSpan=3 vAlign=top>
```

```
    <P><TEXTAREA cols=58 name="yourmessage" rows=10>
    <%=yourmessage%>
    </TEXTAREA> <BR><FONT face=Arial
    size=2>Before you press this button, make sure your e-mail address is 100%
    correct. <B><BR>
    <INPUT type=submit value="Send e-mail" id=submit1 name=submit1>
    <INPUT name=reset type=reset value="Clear forms">
</B></FONT></P></TD></TR></TBODY></TABLE>
</FORM></center>




<%
        elseif flag="complete" then
        Response.Redirect "default.asp"
        end if
%>


<BR>

<center><!--#include file="banners.asp" --></center>


</BODY>
</HTML>
```

Filename: memberlogin.asp

```
<%@ Language=VBScript %>

<!--#include file="functionincludes.asp"-->

<%

Response.Buffer=true
dim myusername, mypassword

'if user hits the login button
if Request("btnLogin") <> "" then

        myusername = trim(Request.Form("username"))
        mypassword = trim(Request.Form("userpassword"))

        if myusername<>"" and mypassword<>"" then

                'call the form function
                if isLegaluser(myusername, mypassword) then

                        'to make sure that the user is valid
                        Session("isvaliduser") = true
                        Response.Redirect "redirector.asp"

                else

                        Session("isvaliduser") = false
                        invalid = "Please re-enter Username and Password"

                end if

        end if
end if

if Request("btnNewuser") <> "" then

        Response.Redirect "newmembers.asp"

end if



%>


<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>

<BODY bgcolor="ghostwhite">
<center><!--#include file="useridbox.asp" -->
</center>
<font face="Arial, Helvetica, sans-serif" size="2">
<BR>
<font color='#ff0000'><%=invalid%></font>
```

```
</font>

<FORM action="memberlogin.asp" id=FORM1 method=post name=FORM1>
<center>
        <TABLE  align="center" width="600" border="0" cellspacing="0" cellpadding="5"
         bgColor=white background="" style="WIDTH: 600px">
 <TBODY>
                <TR>
                        <TD width="100"></TD>
                        <TD width="185"></TD>
                        <TD width="85"></TD>
                </TR>
                <TR>
                        <TD colspan="3">
                        <font face="Verdana,Arial,Helvetica" size="2" color="#000000">
                        <STRONG>Welcome
                        </STRONG>
                        to our AsiStore. We are here to
                        sell goods of your choice
                        for much lesser than anybody else.
                        </font>
                        </TD>
                </TR>
                <TR>
                        <TD colspan="3">
                        <font face="Verdana,Arial,Helvetica" size="2" color="#000000">
                        Please enter your registered AsiStore membership
                         username and your password. 
                        If you are not a registered user please click on the <STRONG>'New
                        Users!</STRONG>


                        ' button.  If you are already a member,simply click on
                         the<b>'Log Me In!'</b>
                        button. <BR><BR>Have fun and Enjoy your shopping at
                        AsiStore.</font></TD>
                </TR>
                <TR>
                        <TD height="20" align="middle" colspan="4">
                        <IMG align=absMiddle alt="" border=0 height=20 src="../images/bar.gif"
                         width=600>
                        </TD>
                </TR>
                <TR>
                        <TD align="right" valign="top">
                        <font face="Verdana,Arial,Helvetica" size="2" color="black">
                        Your Username: 
                        </font></TD>
                        <TD valign="top">
                        <input name="username" value="<%=myusername%>" size="35"
                         maxlength="45">
                        <br>
                        <font face="Verdana,Arial,Helvetica" size="1" color="black">Please use
your
                        complete username
                        <BR>assigned by the systems administrator.
                        <br>
                        </font>
```

```
                    </TD>
                    <td> </td>
                    </TR>
                    <TR>
                            <TD colspan="3"> </TD>
                    </TR>
                    <TR>
                            <TD align="right" valign="top">
                            <font face="Verdana,Arial,Helvetica" size="2" color="black">
                            Your Password: 
                            </font> </TD>
                            <TD>
                            <input type="password" name="userpassword"
                             value ="<%=mypassword%>"
                             size="35" maxlength="20">
                            <br><FONT face=Verdana size=1>Your password has
                            to be more than six<BR>characters in
                            length. 
                            If you have forgotten<BR>your password contact
the
                            systems<BR>
                            administrator for <STRONG>asiStore</STRONG>.<BR> 
                            </FONT></TD>
                            <td> </td>
                    </TR>
                    <TR>
                            <TD colspan="3">    </TD>
                    </TR>
                    <TR>
                      <TD colspan="3" height="10"> </TD>
                    </TR>
                    <TR>

                            <TD colspan="3" align="middle"><a name="login"></a>
                            <input type="submit" name="btnNewuser" value="New Users!">
                            <input type="submit" name="btnLogin" value="Log Me In!">
</TD>
                    </TR>
                    <TR>
                      <TD colspan="3" height="15"> </TD>
                    </TR>
                    <TR>
                            <TD colspan="3"> </TD>
                    </TR>
                    <TR>
                            <TD colspan="3"></TD>
                    </TR>
                    <TR>
                      <TD colspan="3"> </TD>
                    </TR></FORM></TBODY></TABLE></CENTER>
                    <BR>

</BODY>
</HTML>
```

Filename: newmember.asp

```
<%@ Language=VBScript %>

<!--#include file="functionincludes.asp"-->

<%

Response.Buffer=true
dim myusername, mypassword

'if user hits the login button
if Request("btnAdduser") <> "" then

        myusername = trim(Request.Form("username"))
        mypassword = trim(Request.Form("userpassword"))
        remypassword = trim(Request.Form("reuserpassword"))

        if myusername<>"" and mypassword<>"" and remypassword<>"" then
                error_code = validateNewpassword(mypassword, remypassword)
                if error_code = "valid" then
                        myvalue = addNewmember(myusername, mypassword)
                        if myvalue = false then
                                error_code = "<BR>Your username is already in use.
                                 Please Re-enter!!"
                        else
                                Response.Redirect "memberlogin.asp"
                        end if

                end if

        end if
end if



%>


<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>

<BODY bgcolor="ghostwhite">
<center><!--#include file="useridbox.asp" -->
</center>
<font face="Arial, Helvetica, sans-serif" size="2">
<BR>
<font color='#ff0000'><%=error_code%></font>
</font>

<FORM action="newmembers.asp" id=FORM1 method=post name=FORM1>
<center>
        <TABLE  align="center" width="600" border="0" cellspacing="0" cellpadding="5"
        bgColor=white background="" style="WIDTH: 600px">
 <TBODY>
                <TR>
```

```
        <TD width="100"></TD>
        <TD width="185"></TD>
        <TD width="85"></TD>
</TR>
<TR>
        <TD colspan="3">
        <font face="Verdana,Arial,Helvetica" size="2" color="#000000">
        <STRONG>Welcome
         </STRONG>
         to our AsiStore registration page. 
         We sell goods of your choice
        for much lesser than anybody else.
        </font>
        </TD>
        </TR>

        <TR>
        <TD colspan="3">
        <font face="Verdana,Arial,Helvetica" size="2" color="#000000">
        Please  enter a
        name you wish to use as your username
        and a password you wish to use as your secret
        code.  Remember your username and password for
         future visit to AsiStore 
        Do not forget to reconfirm your password.  Then, simply click on
        the<b>'Add me in!'</b> button.
        <BR></font></TD>
        </TR>

        <TR>
                <TD height="20" align="middle" colspan="4">
                <IMG align=absMiddle alt="" border=0 height=20
                src="../images/bar.gif" width=600></TD>
        </TR>

        <TR>
                <TD align="right" valign="top">
                <font face="Verdana,Arial,Helvetica" size="2" color="black">
                Your Username: 
                </font></TD>
                <TD valign="top">
                <input name="username" value="<%=myusername%>"
                size="35" maxlength="45" >
                <br>
                <font face="Verdana,Arial,Helvetica" size="1" color="black">
                Please use your complete username
                <BR>assigned by the systems administrator.
                <br>   </font></TD>
                <td> </td>
        </TR>

        <TR>
                <TD colspan="3"> </TD>
        </TR>


        <TR>
                <TD align="right" valign="top">
```

```
                    <font face="Verdana,Arial,Helvetica" size="2" color="black">
                    Your Password: 
                    </font> </TD>
                    <TD>
                    <input type="password" name="userpassword"
                    value ="<%=mypassword%>"
                    size="35" maxlength="20">
                    <br><FONT face=Verdana size=1>Your password has to be
more
                    than six
                    <BR>characters in
                    length. </FONT></TD>
                    <td> </td>
                </TR>

                <TR>
                    <TD colspan="3"> </TD>
                </TR>



                <TR>
                    <TD align="right" valign="top">
                    <font face="Verdana,Arial,Helvetica" size="2" color="black">
                     Re-enter Password: 
                    </font> </TD>
                    <TD>
                    <input type="password" name="reuserpassword"
                     value ="<%=remypassword%>" size="35" maxlength="20">
 
                    <br><FONT face=Verdana size=1>Please Enter password again
                    to reconfirm.</FONT></TD>
                    <td> </td>
                </TR>


                <TR>

                    <TD colspan="3" align="middle"><a name="login"></a>
                    <input type="submit" name="btnAdduser" value="Add me in!">
</TD>
                <TR>
                  <TD colspan="3" height="15"> </TD>
                </TR>
                <TR>
                    <TD colspan="3"> </TD>
                </TR>
                <TR>
                    <TD colspan="3"></TD>
                </TR>
                <TR>
                  <TD colspan="3"> </TD>
                </TR></FORM></TBODY></TABLE></CENTER>
                <BR>


</BODY>
</HTML>
```

Filename: readuserinformation.asp

```
<%@ Language=VBScript %>
<!--#include file="functionincludes.asp"-->
<%
'check if the user is logged in
   Response.Buffer = true
   Session("counter") = 0
   showsummaryinfo = "no"
   'errormessage = "pk" 'initially no error message
if hasaccessRight = false then

        Session.Abandon
        Response.Redirect "memberlogin.asp"

end if


if Request("submitinfo.X") <> "" then
        Session("counter") = Session("counter") + 1
        'read the user input informations
        nameofuser = Request("txtname")
        if nameofuser ="" then
                errormessage = errormessage & isBlankmessage("Name")
        end if

        emailaddress = Request("txtemail")
        if emailaddress ="" then
                errormessage = errormessage & isBlankmessage("e-mail address")
        else
                errormessage = errormessage & ValidEmail(emailaddress)
        end if

        phone1 = Request("txtphone1")
        if phone1 ="" then
                errormessage = errormessage & isBlankmessage("Area Code ")
        else
                errormessage = errormessage & Isnumericform("Area Code ", "phone1", phone1)
        end if

        phone2 = Request("txtphone2")
        if phone2 ="" then
          errormessage = errormessage & isBlankmessage("Local Code")
        else
         errormessage = errormessage & Isnumericform("Local Code ", "phone2", phone2)
        end if

        phone3 = Request("txtphone3")
        if phone3 = "" then
                errormessage = errormessage & isBlankmessage("extension")
        else
                errormessage = errormessage &
                Isnumericform("extension ", "phone3", phone3)
        end if
        userphone = phone1 & phone2 & phone3

        usermailingaddress = Request("txtaddress")
        if usermailingaddress = "" then
```

```
                errormessage = errormessage & isBlankmessage("Mailing Address")
        end if

        usercity = Request("txtcity")
        if usercity = "" then
                errormessage = errormessage & isBlankmessage("City")
        end if

        userzip = Request("txtzip")
        if userzip = "" then
                errormessage = errormessage & isBlankmessage("Zip Code")
        else
                'errormessage = errormessage & Isnumericform("Zip Code ", userzip)
                errormessage = errormessage & validateStringitem(userzip, "zipcode")
        end if

        stateselected = Request("State")
        if stateselected="" then
        'bydeafult
                stateselected="AL"
        end if
        usercountry = Request("txtcountry")
        if usercountry = "" then
                errormessage = errormessage & isBlankmessage("Country")
        end if

        cardtype = Request("cardtype")
        if cardtype = "" then
                cardtype = "visa"
        end if

        cardnumber = Request("txtcardnumber")

        if cardnumber = "" then
                errormessage = errormessage & isBlankmessage("Credit Card Number")
        else
                errormessage = errormessage & validateStringitem(cardnumber,
"creditcardnumber")
        end if

        expirationmonth = Request("expmonth")
        if expirationmonth="" then
                'bydeafult
                expirationmonth="01"
        end if
        expirationyear = Request("expyear")
        if expirationyear = "" then
                'bydeafult
                expirationyear = "2002"
        end if
        end if


if errormessage = ""  then

        if Session("counter") <> 0 then
                showsummaryinfo = "yes"
        end if
```

```
end if


%>


<html>
<body bgcolor="ghostwhite">



<center>
<!--#include file="useridbox.asp" -->
</center>

<i><font face="Arial, Helvetica, sans-serif" size="2">
<font color="#FF0000"><%=errormessage%></font>
</font></i>

<form name="shop" action="readuserinfo.asp">


<% if showsummaryinfo <> "yes" then  %>

<table align="center" cellpadding="4" width="600">


        <tr bgcolor="#0066ff">
          <td colspan="5"> <font face="Arial, Helvetica, sans-serif" size="2">
            <b><font size="4" color="#ffffff">Billing Information<font size="1">
            </font></font></b></font></td>
        </tr>


        <tr bgcolor="#cccccc">
          <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
          <b>Name </b></font></td>
          <td colspan="3" width="494">
            <input name="txtname" value="<%=nameofuser%>" size="40">
          </td>
        </tr>


        <tr bgcolor="#dcdcdc">
          <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
          <b>E-mail </b></font></td>
          <td colspan="3" width="494">
            <input name="txtemail" value="<%=emailaddress%>" size="40">
          </td>
        </tr>


        <tr bgcolor="#cccccc">
          <td colspan="2" align="right">
          <font face="Arial, Helvetica, sans-serif" size="2">
          <b>Phone </b></font></td>
```

```
 <td colspan="3" width="494">
 <font size="4">
   (<input name="txtphone1" value="<%=phone1%>" maxlength="3" size="3">)
 </font>
   <input name="txtphone2" value="<%=phone2%>" maxlength="3" size="3"> <B>-</B>
    <input name="txtphone3" value="<%=phone3%>" maxlength="4" size="4">
 </td>
</tr>



<tr bgcolor="#dcdcdc">
 <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
 <b>Address </b></font></td>
 <td colspan="3" width="494">
   <input name="txtaddress" value="<%=usermailingaddress%>" size="40">
 </td>
</tr>

<tr bgcolor="#cccccc">
 <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
 <b>City </b></font></td>
 <td colspan="3" width="494">
   <input name="txtcity" value="<%=usercity%>" size="40">
 </td>
</tr>

<tr bgcolor="#dcdcdc">
 <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
 <b>State/Province </b></font></td>
 <td colspan="3" width="494">
 <select name="State">
   <option <%=isSelected(stateselected,"AL")%> value="AL">Alabama</option>
   <option <%=isSelected(stateselected,"AA")%> value="AA">Alaska</option>
   <option <%=isSelected(stateselected,"AZ")%> value="AZ">Arizona</option>
   <option <%=isSelected(stateselected,"AK")%> value="AK">Arkansas</option>
   <option <%=isSelected(stateselected,"CA")%> value="CA">California</option>
   <option <%=isSelected(stateselected,"CO")%> value="CO">Colorado</option>
   <option <%=isSelected(stateselected,"CT")%> value="CT">Connecticut</option>
   <option <%=isSelected(stateselected,"DE")%> value="DE">Delaware</option>
   <option <%=isSelected(stateselected,"DC")%> value="DC">District of Columbia
</option>
   <option <%=isSelected(stateselected,"FL")%> value="FL">Florida</option>
   <option <%=isSelected(stateselected,"GA")%> value="GA">Georgia</option>
   <option <%=isSelected(stateselected,"HI")%> value="HI">Hawaii</option>
   <option <%=isSelected(stateselected,"ID")%> value="ID">Idaho</option>
   <option <%=isSelected(stateselected,"IL")%> value="IL">Illinois</option>
   <option <%=isSelected(stateselected,"IN")%> value="IN">Indiana</option>
   <option <%=isSelected(stateselected,"IA")%> value="IA">Iowa</option>
   <option <%=isSelected(stateselected,"KS")%> value="KS">Kansas</option>
   <option <%=isSelected(stateselected,"KY")%> value="KY">Kentucky</option>
   <option <%=isSelected(stateselected,"LA")%> value="LA">Louisiana</option>
   <option <%=isSelected(stateselected,"ME")%> value="ME">Maine</option>
   <option <%=isSelected(stateselected,"MD")%> value="MD">Maryland</option>
   <option <%=isSelected(stateselected,"MA")%> value="MA">Massachusetts</option>
   <option <%=isSelected(stateselected,"MI")%> value="MI">Michigan</option>
   <option <%=isSelected(stateselected,"MN")%> value="MN">Minnesota</option>
   <option <%=isSelected(stateselected,"MS")%> value="MS">Mississippi</option>
```

```
   <option <%=isSelected(stateselected,"MO")%> value="MO">Missouri </option>
   <option <%=isSelected(stateselected,"MT")%> value="MT">Montana </option>
   <option <%=isSelected(stateselected,"NE")%> value="NE">Nebraska</option>
   <option <%=isSelected(stateselected,"NV")%> value="NV">Nevada</option>
   <option <%=isSelected(stateselected,"NH")%> value="NH">New Hampshire</option>
   <option <%=isSelected(stateselected,"NJ")%> value="NJ">New Jersey</option>
   <option <%=isSelected(stateselected,"NM")%> value="NM">New Mexico</option>
   <option <%=isSelected(stateselected,"NY")%> value="NY">New York</option>
   <option <%=isSelected(stateselected,"NC")%> value="NC">North Carolina</option>
   <option <%=isSelected(stateselected,"ND")%> value="ND">North Dakota</option>
   <option <%=isSelected(stateselected,"OH")%> value="OH">Ohio</option>
   <option <%=isSelected(stateselected,"OK")%> value="OK">Oklahoma</option>
   <option <%=isSelected(stateselected,"OR")%> value="OR">Oregon</option>
   <option <%=isSelected(stateselected,"PA")%> value="PA">Pennsylvania</option>
   <option <%=isSelected(stateselected,"RI")%> value="RI">Rhode Island</option>
   <option <%=isSelected(stateselected,"SC")%> value="SC">South Carolina</option>
   <option <%=isSelected(stateselected,"SD")%> value="SD">South Dakota</option>
   <option <%=isSelected(stateselected,"TN")%> value="TN">Tennessee</option>
   <option <%=isSelected(stateselected,"TX")%> value="TX">Texas</option>
   <option <%=isSelected(stateselected,"UT")%> value="UT">Utah</option>
   <option <%=isSelected(stateselected,"VT")%> value="VT">Vermont</option>
   <option <%=isSelected(stateselected,"VA")%> value="VA">Virginia</option>
   <option <%=isSelected(stateselected,"WA")%> value="WA">Washington</option>
   <option <%=isSelected(stateselected,"WV")%> value="WV">West Virginia</option>
   <option <%=isSelected(stateselected,"WI")%> value="WI">Wisconsin</option>
   <option <%=isSelected(stateselected,"WY")%> value="WY">Wyoming</option>
</select>
  </td>
</tr>

<tr bgcolor="#cccccc">
  <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
  <b>Zip/Postal
    Code </b></font></td>
  <td colspan="3" width="494">
    <input name="txtzip" value="<%=userzip%>" maxlength="5" size="5">
  </td>
</tr>

<tr bgcolor="#dcdcdc">
  <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
  <b>Country
    </b></font> </td>
  <td colspan="3" width="494">
    <input name="txtcountry" size="16" value="USA">
  </td>
</tr>


<tr bgcolor="#cccccc">
  <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
  <b>Card Type
    </b></font> </td>
  <td colspan="3" width="494">
   <select name="cardtype">
      <option <%=isSelected(cardtype,"visa")%> value="visa">Visa</option>
      <option <%=isSelected(cardtype,"discover")%> value="discover">Discover</option>
      <option <%=isSelected(cardtype,"mastercard")%> value="mastercard">Master Card
```

```
</option>
            <option <%=isSelected(cardtype,"americanexp")%> value="americanexp">American
Express
          </option>
            </select>
           </td>
          </tr>


          <tr bgcolor="#dcdcdc">
           <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
           <b>Card Number
            </b></font> </td>
           <td colspan="3" width="494">
            <input name="txtcardnumber" value="<%=cardnumber%>" maxlength="16" size="16">
           </td>
          </tr>

          <tr bgcolor="#cccccc">
           <td colspan="2" align="right"><font face="Arial, Helvetica, sans-serif" size="2">
           <b>
             Expiration
            </b></font> </td>

           <td colspan="3" width="494">
           Month:
            <select name="expmonth">


     <option <%=isSelected(expirationmonth,"01")%> value="01">January</option>
             <option <%=isSelected(expirationmonth,"02")%> value="02">February</option>
             <option <%=isSelected(expirationmonth,"03")%> value="03">March</option>
             <option <%=isSelected(expirationmonth,"04")%> value="04">April</option>
             <option <%=isSelected(expirationmonth,"05")%> value="05">May</option>
             <option <%=isSelected(expirationmonth,"06")%> value="06">June</option>
             <option <%=isSelected(expirationmonth,"07")%> value="07">July</option>
             <option <%=isSelected(expirationmonth,"08")%> value="08">August</option>
             <option <%=isSelected(expirationmonth,"09")%> value="09">September</option>
             <option <%=isSelected(expirationmonth,"10")%> value="10">October</option>
             <option <%=isSelected(expirationmonth,"11")%> value="11">November</option>
             <option <%=isSelected(expirationmonth,"12")%> value="12">December</option>
            </select>
            Year:
            <select name="expyear">
             <option <%=isSelected(expirationyear,"2001")%> value="2001">2001</option>
             <option <%=isSelected(expirationyear,"2002")%> value="2002">2002</option>
             <option <%=isSelected(expirationyear,"2003")%> value="2003">2003</option>
             <option <%=isSelected(expirationyear,"2004")%> value="2004">2004</option>
             <option <%=isSelected(expirationyear,"2005")%> value="2005">2005</option>
             <option <%=isSelected(expirationyear,"2006")%> value="2006">2006</option>

</select>
          </td>
         </tr>
 </table>


<br>
```

```
<table align="center" border="0" cellspacing="0" width="600">
  <tr>
    <td align="center" valign="top"><font color="#000066" size="4" face="Verdana,
        Arial, Helvetica">
    <input TYPE="IMAGE" NAME="submitinfo" value="1012" SRC="../images/submitorder.gif"
        WIDTH="150" HEIGHT="16">
    </td>
  </tr>
</table>
<%      else  %>

<center>
 <table bgcolor="white" border=1 cellpadding=0 cellspacing=0 width=600>
<tr bgcolor=white>

<td align="middle">
<BR>
<BR>
<font color=#6f6f6f size=5>
Thank you for Shopping with us 
</font>
<br>

<font color=#6f6f6f size=2>
Your order will be shipped at the following address
<B>
    <BR><%=nameofuser%>
    <BR><%=usermailingaddress%>
    <BR><%=usercity%>, <%=stateselected%> <%=userzip%>
    <BR>Phone: (<%=phone1%>) <%=phone2%>-<%=phone3%>
    <BR> <B>Order ID: XZWERQWS</B>
          <BR>
          <BR>
         <a href="default.asp"> asiStore </a>
</B>
</font></FONT>
<BR>
<BR>
</td>
</tr>

</table>
 </center>

 <BR>
 <BR>

 <%
 'bulid message string and send confirmation email to the costumer
 messageString = "Thank you for Shopping with and you shipment will be sent at "
 messageString = " " & messageString & nameofuser & ","
 messageString = " " & messageString & " " & usermailingaddress & ","
 messageString = " " & messageString & " " & usercity & "," & stateselected & " " & userzip
 confirm = sendemail (messageString, "pna2915@rit.edu",emailaddress, "Purchase
confirmation" )

 %>
```

```
<%  end if %>
</form>

<!--#include file="banners.asp" -->

</body>
</html>
```

Filename: summariseorder.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">

  <HTML>
        <HEAD>
        </HEAD>

        <BODY bgcolor="ghostwhite">


    <BR/>
    <BR/>


    <TABLE align="center" bgcolor="#0066FF" width="600">
          <tr>

                <td valign="top" bgcolor="#0066FF">
                <font color="white">
                <B>Product ID</B>
                </font>
                </td>

                <td valign="top" bgcolor="#0066FF">
                <font color="white">
                <B>Product Name</B>
                </font>
                </td>

                <td valign="top" bgcolor="#0066FF">
                <font color="white">
                <B>Units</B>
                </font>
                </td>

                <td valign="top" bgcolor="#0066FF">
                <font color="white">
                <B>Unit Price</B>
                </font>
                </td>

                <td valign="top" bgcolor="#0066FF">
                <font color="white">
                <B>Unit Total</B>
                </font>
                </td>

        </tr>

      <xsl:for-each select="Products/Product">

        <TR>
        <xsl:attribute name="STYLE">
        background-color:<xsl:eval>whichColor(this)</xsl:eval>
        </xsl:attribute>
```

```
        <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
          <xsl:value-of select="ProductID"/>
        </TD>

        <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
          <xsl:value-of select="ProductName"/>
        </TD>

        <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
          <xsl:value-of select="Units"/>
        </TD>

        <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt">
          $<xsl:value-of select="Price"/>
        </TD>

        <TD STYLE="font-family:Arial, sans-serif; font-size: 10pt"> <!-- line total -->
          <xsl:eval>formatNumber(lineTotal(this), "$#,##0.00")</xsl:eval>
        </TD>

      </TR>

    </xsl:for-each>

            <xsl:for-each select="Products">

      <TR>
      <TD COLSPAN="3"></TD>

      <TD STYLE="font-family:Arial, sans-serif; font-size: 11pt">
       <B><font color="white">
        Total Order:
       </font></B>
      </TD>

      <TD STYLE="text-align:left; border:none; border-top:1px solid black">
       <B>
       <font color="white">
        <xsl:eval>formatNumber(invoiceTotal(this), "$#,##0.00")</xsl:eval>
       </font>
       </B>
      </TD>
     </TR>
    </xsl:for-each>

  </TABLE>


<BR/>

<table align="center" width="600">

      <TR>
      <TD align="center"><a href ="cart.asp">
      <img border="0" src="../images/changeorder.gif"></img>
      </a>
```

```
        </TD>
        </TR>

        <TR>
        <TD align="center">
        <a href = "readuserinfo.asp">
        <img border="0" src="../images/placeorder.gif">
        </img>
        </a>
        </TD>
        </TR>

        </table>

  </BODY>
  </HTML>
</xsl:template>


<xsl:script><![CDATA[
  function invoiceTotal(invoice)
  {
    //this function sums of the total amount of Money Spent
    items = invoice.selectNodes("Product");
    var sum = 0;

    for (var item = items.nextNode(); item; item = items.nextNode())
    {
      var price = item.selectSingleNode("Price").nodeTypedValue;
      var qty = item.selectSingleNode("Units").nodeTypedValue;
      sum += price * qty;
    }
    return sum;
  }

  function lineTotal(item)
  {
    //this function will calculate the Unit Total (unitTotal = price * Units)
    var price = item.selectSingleNode("Price").nodeTypedValue;
    var qty = item.selectSingleNode("Units").nodeTypedValue;
    return qty*price;
  }

  function even(e) {
    return childNumber(e) % 2;
  }

  function whichColor(e) {
    if (even(e))
      return "#efefef";
    else
      return "#ffffff";
  }


 ]]></xsl:script>

</xsl:stylesheet>
```

Filename: transferserverside.asp

```
<%@ Language=VBScript %>

<%
  'compiling the two files on the server XML and XSL
  ' and throwing it to the browser
  'this one netscape anybrowser will recognize it

   Response.Buffer = true
        Set xmldoc = CreateObject("Msxml2.DOMDocument")
        Set xmlStyle = CreateObject("Msxml2.DOMDocument")

        'set flags
        xmldoc.async = false
        xmlStyle.async = false

        xmldoc.validateOnParse = false
        xmlStyle.validateOnParse = false

        'load the xml document
        xmldoc.load(Server.MapPath("products.xml"))
        xmlStyle.load(Server.MapPath("summariseorder.xsl"))


        'if ((xmldoc.parseError == 0) && (xmlStyle.parseError == 0)) then
                Response.Write xmldoc.transformNode(xmlStyle)
        'else if
           'Response.Write "<BR>error returning objects"
        'end if


  %>
```

Filename : validateuser.asp

```
<%@ Language=VBScript %>
<!--#include file="functionincludes.asp"-->
<%

Response.Buffer=true
dim myusername, mypassword

myusername = trim(Request.Form("username"))
mypassword = trim(Request.Form("userpassword"))

if myusername<>"" and mypassword<>"" then
        'call the form function
        if isLegaluser(myusername, mypassword) then
                'to make sure that the user is valid
                Session("isvaliduser") = true
                Response.Redirect "redirector.asp"
        else
                Session("isvaliduser") = false
                invalid = "Please re-enter Username and Password"
        end if
end if

%>


<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
</HEAD>

<BODY>


<table border="0" cellspacing="0" width="600">
  <tr>
    <td valign="top"><font color="#000066" size="4" face="Verdana, Arial, Helvetica">
      <b>Welcome! </b></font><font
    color="#000066" size="2" face="Verdana, Arial, Helvetica">
      <b><%=FormatDateTime(now(),vblongdate)%></b></font></td>
  </tr>
</table>

<font face="Arial, Helvetica, sans-serif" size="2">
<font color='#FF0000'><%=invalid%></font>
</font>

<FORM action="validateuser.asp" id=FORM1 method=post name=FORM1>


 <div align="center">

  <table width="300" bgcolor="darkgray">

    <tr>
      <td align="right">Username</td>
      <td colspan="3">
```

```
      <input value="<%=myusername%>" id="text1"
        name="username" maxlength="10" size="15">
      </td>
    </tr>

    <tr>
      <td align="right">Password</td>
      <td colspan="3">
        <input value = "<%=mypassword%>" id="password1" name="userpassword"
          type=password maxlength=15 size=15>
      </td>
    </tr>

    <tr align="center">
      <td colspan="4">
        <input id=submit1 name=btnlogin type=submit value=login>
      </td>
    </tr>

  </table>
  <BR>
  <BR>
 </div>
</FORM><BR>

</BODY>
</HTML>
```

Source code pullxmlfromdll Visual Basic component

```
Option Explicit
Dim xmlDomDoc As New MSXML2.DOMDocument
Dim pid As String

'*************************************************************************************************
' Function: Load_File()
' Purpose:  Loads Template file
' Parameters: no parameter is passed
' Return Values: none
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************


Public Function Load_File()
   'Set the mode to asynchronous ie control will return only
   'if the entire document has been loaded
   xmlDomDoc.async = False
   'Load XML document into DOMDocument object
   xmlDomDoc.Load App.Path & "/Products.xml"
   Load_File = xmlDomDoc.xml  'loading successful
End Function

'*************************************************************************************************
' Function: Load_String
' Purpose:  Loads Template file
' Parameters: xml string passed
' Return Values: will return the loaded xml string
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************

Public Function Load_String(xmlString)
   'will load the xml string
   xmlDomDoc.async = False
   xmlDomDoc.loadXML xmlString
   Load_String = xmlDomDoc.xml
End Function

'*************************************************************************************************
' Function: addItem
' Purpose:  This item will add new item into the basket
' Parameters:
' Return Values:
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************


Public Function addItem(pid As Variant, productsname As Variant, productsprice
```

```
As Variant, productunits As Variant)
  'Each <Product> element in XML document contains four child elements
  'ie <ProductID>,<ProductName>,<Price> and <Units>
  Dim Product As IXMLDOMNode
  Dim ProductID As IXMLDOMNode, productname As IXMLDOMNode,
  price As IXMLDOMNode, units As IXMLDOMNode

  On Error Resume Next

  'Create <Product></Product> node


    Set Product = xmlDomDoc.createNode(1, "Product", "")

    'Create <ProductID></ProductID> node
    Set ProductID = xmlDomDoc.createNode(1, "ProductID", "")
    ProductID.Text = pid

    'Create <ProductName></ProductName> node
    Set productname = xmlDomDoc.createNode(1, "ProductName", "")
    productname.Text = productsname

    'Create <Units></Units> node
    Set units = xmlDomDoc.createNode(1, "Units", "")
    units.Text = productunits

    'Create <Price></Price> node
    Set price = xmlDomDoc.createNode(1, "Price", "")
    price.Text = productsprice

    'Append <ProductID>,<ProductName>,<Price> and <Units> as children
    'to <Product> node
    Product.appendChild ProductID
    Product.appendChild productname
    Product.appendChild units
     Product.appendChild price
    xmlDomDoc.documentElement.appendChild Product
    addItem = xmlDomDoc.xml

  'Add the <Product> node to XML tree and save
  'xmlDomDoc.save App.Path & "/Products.xml"
  'Reset the text boxes

End Function

'*************************************************************************************************
' Function: getItem
' Purpose:  Loads Template file
' Parameters:
' Return Values: returns true of founfd false if not found
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************

Public Function getItem(pid As Variant)
  Dim modNode As IXMLDOMNode
```

```
    'Get the node with the ProductID entered by user
    Set modNode = xmlDomDoc.selectSingleNode("Products/Product[ProductID='" & pid & "']")
    If modNode Is Nothing Then
       getItem = False
       'Display error message if the ProductID does not exist
    Else
       getItem = True
    End If
End Function

'*****************************************************************************************************
' Function: updateItem
' Purpose:  Loads Template file
' Parameters:
' Return Values:
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*****************************************************************************************************

Public Function updateItem(pid As Variant, productsunits As Variant)
    'Each <Product> element in XML document contains four child elements
    'ie <ProductID>,<ProductName>,<Price> and <Units>
    Dim newChild As IXMLDOMNode
    Dim oldChild As IXMLDOMNode
    Dim ProductID As IXMLDOMNode, productname As IXMLDOMNode,
   price As IXMLDOMNode, units As IXMLDOMNode

    On Error Resume Next

    'initialize the oldchild

    Set oldChild = xmlDomDoc.selectSingleNode("Products/Product[ProductID='" & pid & "']")
    'Create <Product></Product> node
    Set newChild = xmlDomDoc.createNode(1, "Product", "")

    'Create <ProductID></ProductID> node
    Set ProductID = xmlDomDoc.createNode(1, "ProductID", "")
    ProductID.Text = oldChild.childNodes.Item(0).Text

    'Create <ProductName></ProductName> node
    Set productname = xmlDomDoc.createNode(1, "ProductName", "")
    productname.Text = oldChild.childNodes.Item(1).Text


    'Create <Units></Units> node
    'this is the unit which will be updated
    Set units = xmlDomDoc.createNode(1, "Units", "")
    units.Text = productsunits

    'Create <Price></Price> node
    Set price = xmlDomDoc.createNode(1, "Price", "")
    price.Text = oldChild.childNodes.Item(3).Text

    'Append <ProductID>,<ProductName>,<Units> and <Price> as children
    'to <Product> node
    newChild.appendChild ProductID
```

```
    newChild.appendChild productname
    newChild.appendChild units
    newChild.appendChild price

    'Get the <Product> node to be replaced
    'Set oldChild = xmlDomDoc.selectSingleNode("Products/Product[ProductID='" & pid & "']")

    'Replace old <Product> node with new <Product> node
    xmlDomDoc.documentElement.replaceChild newChild, oldChild
    'Save changes to XML file
    'xmlDomDoc.save App.Path & "/Products.xml"
    pid = ""
    updateItem = xmlDomDoc.xml
End Function


'*************************************************************************************************
' Function: deleteItem
' Purpose:  Loads Template file
' Parameters:
' Return Values:
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************

Public Function deleteItem(pid As Variant)
    Dim delNode As IXMLDOMNode

    'Get the node with the ProductID entered by user
    Set delNode = xmlDomDoc.selectSingleNode("Products/Product[ProductID='" & pid & "']")

    If delNode Is Nothing Then
        'Display error message if the ProductID does not exist
        deleteItem = False
    Else
        'Get the values for the selected ProductID into text boxes
        xmlDomDoc.documentElement.removeChild delNode
        'xmlDomDoc.save App.Path & "/Products.xml"
        'txtXML.Text = xmlDomDoc.xml
        deleteItem = xmlDomDoc.xml
    End If
End Function


'*************************************************************************************************
' Function: returnNumunits(pid As Variant)
' Purpose:  This function will take in the pid and return the number of units
' That has been allocated to that pid
' Parameters: pid
' Return Values: none
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: December 29, 2000
'*************************************************************************************************
```

```
Function returnNumunits(pid As Variant)

    Dim oldChild As IXMLDOMNode
    On Error Resume Next
   'initialize the oldchild

    Set oldChild = xmlDomDoc.selectSingleNode("Products/Product[ProductID='" & pid & "']")

    If oldChild.childNodes.Item(2).Text <> "" Then
      returnNumunits = oldChild.childNodes.Item(2).Text
    Else
      returnNumunits = 0
    End If

End Function


'*************************************************************************************************
' Function: givexmlforDisplay()
' Purpose:  This function will just output the XML string
' Parameters: none
' Return Values: none
' Preconditions: none
' Postconditions: none
' Side Effects: node
' Modified: Febrauary 12, 2000
'*************************************************************************************************

Function givexmlforDisplay()


   givexmlforDisplay = xmlDomDoc.xml


End Function
```

Source code for scanadotoxml Visual Basic Component

```
'Puskar Adhikari
'February 04, 2001
'this module is used to convert the database recordsets into xml String


'*********************************************************************************************
' Function: returnmxlstring
' Purpose:  Takes in database connection and query then returns xml string
' This will be taking in recordsets and converting into XML String
' Parameters: no parameter is passed
' Return Values: none
' Preconditions:
' Postconditions:
' Side Effects:
' Modified: February 04, 2001
'*********************************************************************************************


Public Function returnxmlstring(ByRef myConnection, mySQL As Variant)

   Dim myString As String
   myString = ""

   'execute recordset and the query
   Set RS = myConnection.Execute(mySQL)
   'this will be the root node

   RS.MoveFirst  'move to the first recordset

   While Not RS.EOF
      myString = myString & "<Product>"
         For i = 0 To RS.Fields.Count - 1

            myString = myString & "<" & Trim(RS.Fields(i).Name) & ">"
            myString = myString & Trim(RS.Fields(i))
            myString = myString & "</" & Trim(RS.Fields(i).Name) & ">"
         Next

      myString = myString & "</Product>"
      RS.MoveNext
   Wend

   returnxmlstring = myString
   Set RS = Nothing

End Function
```