

Internetworking with SOAP

Terho Antila

Helsinki University of Technology
T-110.551 Seminar on Internetworking
tantila@cc.hut.fi

Abstract

Web services enable businesses to dynamically exchange various information through common interfaces and well-established protocols. SOAP with its extensions is the most promising one of these protocols. Several technologies have been proposed to extend capabilities of SOAP to meet the wide range of business requirements.

This paper introduces the reader to these technologies and presents the problems they are to solve. Finally a peek to the future is taken in the form of virtual enterprises.

KEYWORDS: SOAP, Web Services, Virtual Enterprises, WS-Routing, WS-Security, WS-Trust

1 Introduction to Web Services

The Internet today is much more than just a big warehouse of information. It can also be seen as a platform through which services are delivered to businesses and customers. [1] Business-to-Business (B2B) and Business-to-Customers (B2C) web services are playing a major role in automating businesses' core processes. B2C services have already become familiar to most of the Internet users. Examples of B2C applications include customized news delivery, traffic monitoring and virtual stores. [2]

A good real-life example of a B2B web service is *GlobalWeather*¹, which offers detailed weather data from all over the world. The data is published as a web service, to which customers – often other businesses – are able to make queries in the form of SOAP messages. Other examples where B2B web services are often used include customer relationship management, billing, accounting and supply chain. In fact, B2B web services can be used wherever communication with other businesses exists in company's business model. [2]

Example. Let us suppose that we have a developer of wages calculation applications (e.g. *SalaryCorp*). Sometimes calculating taxes might get very tricky and it would be in the interests of the developer to outsource the parts of the application where this complicated calculation is done. A true specialist in taxes in Finland is naturally Finnish Tax Administration (FTA). Luckily enough, FTA offers a web service², that is capable of doing tax calculations when given certain base information, such as person's annual income,

taxation type etc. Knowing this, *SalaryCorp* decides to utilize the service provided by FTA by integrating a web service client to its application to handle queries to the service.

Traditionally (without using web services) in the situation like above, FTA would have offered a component to be delivered and installed to *SalaryCorp*'s application. A problem arises however, if the taxation conventions happen to change due to a new law: FTA would have to publish a new version of the component and clients would have to install it all over again. When using web services, no action needs to be taken by *SalaryCorp*, but all the modifications are done to FTA's application which is offering the service behind the service interface. Another kind of problem is faced, if *SalaryCorp* wants to change its development platform. They would have to ask FTA for a new implementation of the component – with any luck they have one, most likely not. Among other things web services are designed to offer solutions to this kind of problems.

This paper introduces you to the basic idea behind SOAP and explains the main elements of SOAP architecture in Section 2. Next, in Section 3, features of SOAP are brought up through concepts of security, reliability and scalability. Some examples of latest published proposals to meet these concepts are reviewed in Section 4. Finally, we will take a look at the future through presentation of Virtual Enterprises in Section 5.

2 SOAP

SOAP is a protocol for sending XML messages between computers (or nodes) participating a web service. It was originally planned to be a better way of integrating distributed object technologies – compared to technologies such as CORBA, DCOM and RMI – with native Internet technologies such as XML and HTTP.[3]

SOAP is used for exchanging packages of information between two nodes, sender and receiver. Zero or more nodes can act between these two nodes, which for example convey, re-route and process packages. Each node processes the message if so instructed by the header part of the message. [4]

In the following four subsections, Sections 2.1 - 2.4, SOAP is introduced the same way it is introduced in the specifications by W3C³. The final subsection, Section 2.5, presents two widely accepted technologies around SOAP: WSDL and UDDI.

¹<http://www.capescience.com/webservices/globalweather/index.shtml>

²I have no information whether or not a service like this is actually available at FTA

³World Wide Web Consortium: <http://www.w3.org>

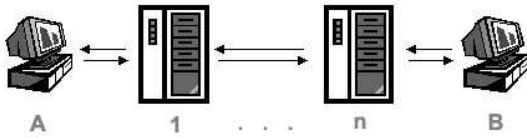


Figure 1: Simple SOAP Architecture. The Initial Sender (A), the Ultimate Receiver (B) and intermediary nodes (1...n)

2.1 Nodes and Roles

The node that initiates the message is called the *Initial SOAP Sender*, whereas the ultimate target node of the message is called the *Ultimate SOAP Receiver*. The nodes acting in the message path are called *Intermediary nodes*. Figure 1 explains the relationship of these nodes. Each node has a certain role according to its relation to the message at hand. Whenever a node receives a message addressed to it, it handles the message according to its role and to instructions given in the header part of the message. Different parts of the message can be indicated to different nodes.

2.2 Message Processing

The entire SOAP message is in XML format. It is constructed from the following parts: *envelope*, *header*, *header block*, *body* and *fault*. The outermost part is the envelope. It is the casing of the message; an XML element that includes two child elements: the header and the body. The header part is a collection of zero or more header blocks, each of which can be targeted at any node within the messages path. The body part is the actual application data sent from the initial sender to the ultimate receiver. The fault part is reserved for fault information that might be generated by any node. The basic structure of a SOAP message is presented in Figure 2.

2.3 Extensibility Model

SOAP in its very core functionality is very basic. The *Extensibility Model* of SOAP can be used to expand the functionality of SOAP messaging framework to satisfy the needs of the desired implementation. The *header* element of a SOAP message can contain any number of child elements each of which is some form of extension to the base SOAP protocol. Perhaps one element contains data associated with the initial sender node and the ultimate receiver node. Another element might contain authentication information, and so on. Some examples of different extensions are introduced in Section 4.

2.4 Binding Framework

The underlying protocol that is used for transporting SOAP messages is not set by W3C's SOAP specification, but a variety of protocols can be used. The *SOAP Protocol Binding Framework* provides general rules for specifying different protocol bindings.

HTTP is definitely the most commonly used protocol for transporting SOAP messages, but others – such as raw TCP or SMTP – can be used as well.

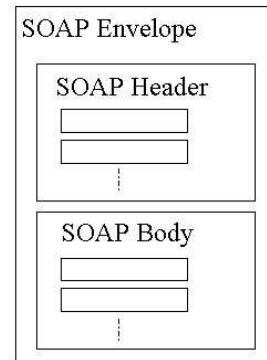


Figure 2: SOAP Message Structure. The message is wrapped in an envelope that can contain several header parts and body parts.

2.5 WSDL and UDDI

Web Services Description Language (WSDL) is a language developed for describing the operational features of web services. It is based on XML and is composed of *interface* and *implementation* definitions. The interface part is an abstract and reusable definition that describes the type of service that is offered. The implementation part describes the way the interface is implemented by certain service provider. WSDL is standardized by W3C⁴. [2] [5] [7]

Universal Description, Discovery, and Integration (UDDI) is a sort of “yellow pages” of web services. It defines a way to publish services (*publication API*) and to make inquiries of published services (*inquiry API*). The information of published services is held in a business registry. [2] [8]

The idea for SOAP, WSDL and UDDI is to inter-operate so, that a client searching for a service fitting to its needs first contacts UDDI to find out an applicable service provider. Next it contacts provider's service and asks for WSDL formatted description of the service interface. The description should offer enough information to make queries to the service. All this can be done – if well organized – dynamically without human interaction. These three technologies together form the basic skeleton for web services. [9]

3 What is SOAP good for?

Figure 3 illustrates how communication between two nodes is done when using one intermediary node. In the picture you can easily see the binding framework, that was introduced in subsection 2.4, in action. In this case SOAP messaging is bound to HTTP, which in turn is implemented on TCP/IP.

The message is constructed and sent by the running application in the sender node and eventually received by the application providing the service. These two applications act as if they were communicating directly to one another. In truth there is one intermediary node passing and presumably processing the messages. And this is the beauty of SOAP!

⁴<http://www.w3.org/2002/ws/desc/>

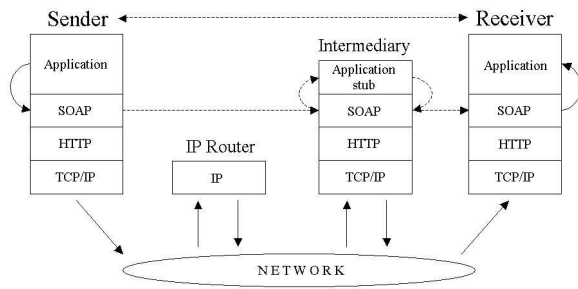


Figure 3: SOAP Routing Network. The sender is communicating with actual receiver through one intermediary node.

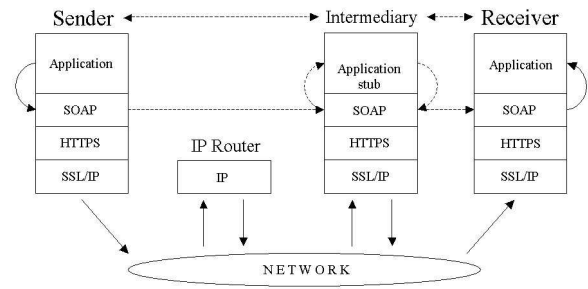


Figure 4: SOAP Security. The sender communicates directly with the intermediary node in belief that he is communicating with the ultimate receiver.

Application level routing enables these “hidden” nodes to act between peer nodes without the peer nodes necessarily knowing anything about the processing done along the path of the message.

A variety of tasks can be carried out by the intermediary node. Since there is a moderated lightweight version of the application (or full version if necessary) running in the node, it can actually look into the application data that is being sent and analyze it. This, along with the message header, allows the node to determine what kind of action should be taken regarding the message. Should it be sent back to the originator? Should the application data be modified and then passed on? Should some third party be informed of the message? Should the message simply be dropped? Should it be re-routed via some other node specialized in particular action? Or should the message plainly be sent forward to the actual receiver?

Answers to these questions depend on the type of the application in question. Some applications may require high security, others good availability and so forth. The following subsections introduce you to some of these characteristics and describe how they can be accomplished. Each feature requires utilizing the SOAP extensibility model or the SOAP binding model, which were discussed in Section 2. One very important aspect – if not the most important one – in network communication is security. The following subsection, Section 3.1 deals with security issues. Section 3.2 introduces a way SOAP can be used to provide improved reliability, availability and scalability. Examples of up-to-date technologies making the most of the extensibility model and offering useful frameworks to promote the issues discussed in this section are presented in Section 4.

3.1 Security

Confidentiality, authentication, integrity and non-repudiation are all elements of computer security. One obvious way of accomplishing these is to use existing secure protocols already out there by making use of the SOAP binding framework. Suppose you replace the HTTP protocol in Figure 3 with HTTPS which in turn uses SSL on top of TCP. All you need to do after this is to get a valid certificate from the service provider (Receiver) and your communication is secured. [10] You must, however, still make it possible for intermediary node(s) to take part

in communication. This might present a problem, since HTTPS is designed for secured peer-to-peer communication only.

Therefore it might be a good idea to divide messaging depicted in Figure 3 into two parts: secure HTTPS-connection from sender to intermediary and another one from intermediary to receiver. Thus original sender interacts with the intermediary node as if it was the ultimate receiver and only needs the certificate from that node. Both parts of the connection are secured and the intermediating node is capable of performing its operations on the data. After this, of course, Figure 3 no longer corresponds to the real situation, which is now presented in Figure 4.

Authentication and encryption of messages can be done at the SOAP level as well. The extra value compared with the previous solutions is the detachment from the transport layer: using SOAP level security abstraction liberates the implementer to use any kind of lower level protocols. Several proposals for web service security abstractions have been published. One proposal for security abstraction is Gordon’s and Pucella’s *Validating a Web Service Security Abstraction by Typing* [11]. Another advantage that application level encryption and authentication has, is the ability to interact with the application data. The intermediaries can for example check, that the initial sender truly is allowed to handle certain data.

3.2 Reliability, Availability and Scalability

Reliability and availability are and have always been two of the major concerns in internetworking: Internet, after all, is an open network whose subnetworks for the most parts are controlled by private corporations. Other than relying on the underlying protocols’ reliability, the reliability of SOAP messaging can also be improved at higher level. This can be accomplished in two ways: *store-and-forward* and *replication of services*. [6]

Store-and-forward is used for boosting reliability and is most likely done in intermediary nodes: the node stores the message it is supposed to convey and forwards it as many times as necessary to reach the next node. This way, if the next node for some reason crashes, it will still receive the message once rebooted. This method of securing reliability is somewhat problematic if some intermediary node will not

come back on-line in fair time. This is true particularly if the initial sender stops and waits until it receives an answer message from the ultimate receiver (which is often the case with RMI-like messaging).

Replicating services is a method to ensure availability. It is often a good idea to have several nodes offering the same service. If one destination node shuts down for some reason, an intermediary node can re-route the message to another destination that offers the same services as the one that became unavailable. An obvious problem is synchronizing: what if the implementation of the service changes? All the replicating services would have to update their implementation instantly to guarantee the integrity of the service.

Replicating services can be also effectively used for *balancing traffic*. All replicas can be hidden behind an application level firewall, who actively polls the network and hardware status and allocates traffic according to some algorithm. The client communicates with the firewall application stub, who then forwards the messages to an appropriate receiver, and vice versa.

Effective scalability is sometimes needed, when the same message needs to be sent to multiple destinations. In this case an intermediary (or the source) node can multiply the message and send each copy to a desired destination. This is certainly an awkward way of redistributing messages – a better way would be to use some sort of Multicast-like convention so that the same message would not be sent to the same node more than once. [12]

3.3 Integration of Layer 8

Open System Interconnection (OSI) Model describes seven layers of abstraction, of which the 7th layer is unofficially called *application layer* – the layer in which SOAP operates. A proposed 8th layer is often described as financial or economic layer.

Use of SOAP extensibility model enables designers to implement for example different kinds of charging methods based on the amount and type of data transmitted. This is a particularly important point of view, when we are talking about *Virtual Enterprises* [13]. Before dynamic web services can really make a major breakthrough, a common practice for dynamic billing must be developed to guarantee the profitability of setting up a web service targeted at multiple and – at the greatest extent – unsettled agile clients. Virtual enterprises are discussed in Section 5.

4 The Sharp Edge of Technology

The methods to improve versatility of web services described in Section 3 need to be abstracted. In this way application developers do not need to “re-invent the wheel”, but are able to take an advantage of the existing technologies and can build the application specific logic on top of them.

*Web Services Interoperability Organization (WS-I)*⁵ is an open consortium of corporations and individuals to promote web services interoperability across platforms, applications,

and programming languages. Several proposals for web service security, routing, trust, transactions etc. have been published by the members of this organization. The most important realizations of these technologies are briefly introduced in the following subsections.⁶

4.1 WS-Routing

The basic SOAP introduced in Section 2 does not define the actual path that the individual messages travel from the initial sender to the ultimate receiver. The idea is, that the underlying protocol to which SOAP is bound, handles the transport of the messages and is therefore also responsible for routing.

However, as shown in Section 3, the ability to accurately specify all the intermediaries through which the message travels on the application level, is essential what it comes to certain implementations of security, reliability and availability.

WS-Routing utilizes the SOAP Extensibility Model and defines a framework for specifying the entire path of a SOAP message. In particular it can be used to define both the forward message path and the reverse message path.

The message path can be dynamically altered during the course of the message: each intermediary may insert additional intermediaries to the message path. This means that the initial sender does not necessarily have to be aware of all the intermediaries through which the message is to be conveyed.

The message path from the initial sender node to the ultimate receiver node is called *forward message path*. In addition to this, an optional *reverse message path* can be used as well. The reverse message path is built dynamically as the message travels on the forward message path. The ultimate receiver node can use this automatically generated message path, if and when replying to the initial sender node. Therefore the use of reverse message path makes it possible to easily use SOAP for two-way communication between peer nodes. [14]

4.2 WS-Security

WS-Security is designed to be a building block helping implementers to enhance message integrity, message confidentiality and single message authentication when building secure web services. It offers a set of SOAP extensions through which these three qualities are to be approached. *WS-Security* is also called *Web Services Security Language*. *WS-Security* builds on *WS-Routing* as it uses certain services for key management and such to which intermediaries are to re-route suitable messages.

Three main mechanisms are provided: *security token propagation*, *message integrity* and *message confidentiality*. These can all be used separately or tightly bound together. These mechanisms do not describe explicit security protocols, but are meant to be used to construct wide range of security protocols. [15]

⁵<http://www.ws-i.org>

⁶*WS-Routing*, *WS-Security* and *WS-Trust* are proposals only, and are not, to my knowledge, in production use

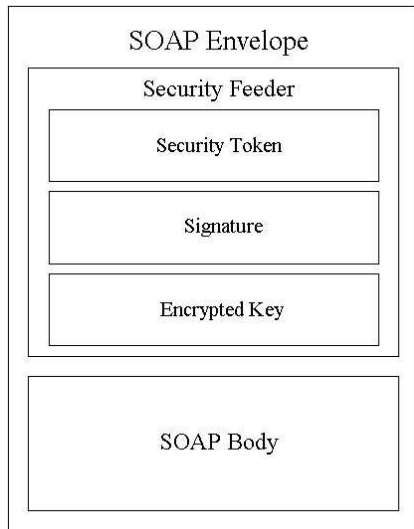


Figure 5: WS-Security Message Structure

Figure 5 illustrates the basic structure of a SOAP message extended with WS-Security. The header called *Security Feeder* includes three elements. *Security Token* is used for authentication, *Signature* for integrity and *Encrypted Key* for confidentiality. [19]

4.3 WS-Trust

Web Services Trust Language (WS-Trust) is used to request and issue security tokens and to manage trust relationships. As WS-Security was built on WS-Routing, WS-Trust is in turn an extension to WS-Security. This is only natural since WS-Security already provides a level of trust. Even though communication between two parties might be “secure” via using WS-Security extension, the question of “trust” remains – can the other party trust the security credentials of the other party? WS-Trust defines extensions to WS-Security providing methods for issuing and exchanging security tokens and ways to establish and access the presence of trust relationships. The goal is to offer businesses a way for trusted SOAP message exchange. [17]

5 A Peek to the Future: Virtual Enterprises

When taken to the extreme, web services enable a business to concentrate on its core process only – everything else is outsourced. The provider company of these external services conducts itself the same way: it only implements the services belonging together with its core process. When this is done recursively, we finally have a supply chain consisting of businesses (“nodes”) using other businesses’ services and advertising its services to others. Each node performs a certain well defined task. Service requests, that do not adjust to node’s core competence, are retransmitted forward to one or more of the service suppliers node uses. When this is done efficiently, company’s costs and time-to-market is reduced,

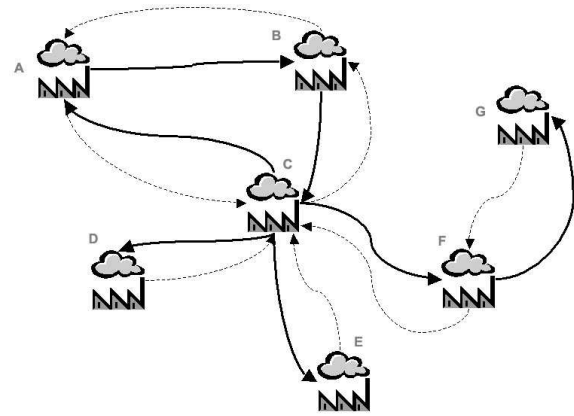


Figure 6: Virtual Enterprises, chain of service requests

while increasing flexibility and diversity of potential service suppliers.

An illustration of the chain of service requests is depicted in Figure 6. Company A initiates the chain of requests by contacting B’s web service. In the figure service requests are characterized by dense heavy arrows, whereas replies (if any) are presented with dotted arrows. In this situation A’s request does not entirely fit into B’s core process. Therefore B uses C’s services to fulfill the request.

C’s core process is focused on extremely narrow field and most of its operation in this chain of requests consists of conveying requests to appropriate suppliers. In fact, C’s business model can be actually based on gathering information on available web services, evaluating them and offering mediating services to others.

Again, F consults services offered by G to fulfill C’s service request, and so on... Eventually B receives a reply from C and is able to fulfill A’s request and answer to it.

Even through this very simplified example, it should be clear by now, that technologies discussed earlier in this paper are necessary and – in many parts – still inadequate. For realization of ultimate virtual enterprises requires certain assumptions to be true:

- All businesses participating this “union of web services” will use standardized descriptions of their core competence and of services they advertise
- The benefits discussed earlier overcome the costs of connection charges and increased delay of service
- Certification services can be trusted
- Contractual agreements can be made dynamically in well defined manner and violators can be located and punished

The mentioned assumptions will be true only, if common standardizations are defined and their execution controlled. [13]

6 Conclusion

Big names, such as Microsoft and IBM, with big intentions, is a combination that has indisputably been giving powerful impulses on development of web technologies. This is the case also with web services, as proposals for secure and inter-operable solutions have been provided, reviewed, enhanced and will eventually be standardized. Will this development lead to robust, secure and inter-operable solutions the industry today needs?

Eventually, yes. Or as Paul Roth from CommerceQuest Inc. so elegantly puts:

“While web services nirvana may still be two to three years away, companies can begin moving toward that goal and gain early business benefits by putting a service-based architecture in place. A service-based architecture allows companies to expose and reuse data as a service and should be considered a secure steppingstone to a full web services implementation.” [18]

It seems that SOAP is the technology web services will be built on also in the future. This should not become as a surprise to the readers of this paper, since the advantages of SOAP have been pointed out throughout the entire paper.

The ultimate virtual enterprises, where a company can dynamically select networks of suppliers to provide exactly what it needs and when it needs it, might be a wet dream of IT enthusiasts. However movement toward loosening coupling between businesses is evident and well justified. SOAP and its extensions offer a suitable framework for implementing virtual enterprises once the common issues and their solutions discussed in Sections 3 and 4 can be agreed on.

7 Acknowledgements

Thanks to Teemu Koponen, the patient tutor.

References

- [1] Dan Jong Kim, Manish Agrawal, Bharat Jayaraman, H. Raghav Rao. A comparison of B2B e-service solutions. Communications of the ACM (Volume 46 Issue 12). Dec. 2003. Available at: ACM Digital Library
- [2] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, A. K. Elmagarmid. Business-to-business interactions: issues and enabling technologies. The VLDB Journal - The International Journal on Very Large Data Bases (Volume 12 Issue 1). May 2003. Available at: ACM Digital Library
- [3] Tim Ewald. The Argument Against SOAP Encoding. Microsoft Corporation. Oct. 2002. [Referenced 15.3.2004]. Available at: <http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx?pull=/library/en-us/dnsoap/html/argsoape.asp>
- [4] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau and Henrik Frystyk Nielsen. SOAP Version 1.2 Part 1: Messaging Framework, W3C. [Referenced: 25.2.2004] Available at: <http://www.w3.org/TR/SOAP/>
- [5] World Wide Web Concoritum. Web Services Description Working Group. [Referenced: 21.3.2004]. Available at: <http://www.w3.org/2002/ws/desc/>
- [6] Rohit Khrane SOAP Routing: The Missing Link, O'Reilly Emerging Technology Conf. (May 2002) [Referenced: 25.2.2004] Available at: <http://www.ics.uci.edu/rohit/ETcon-SOAProuting.ppt>
- [7] Brahim Medjahed, Athman Bouguettaya, Ahmed K. Elmagarmid. Composing Web Services on the Semantic Web. The VLDB Journal - The International Journal on Very Large Data Bases (Volume 12 Issue 4). Nov. 2003. Available at: ACM Digital Library
- [8] Andy Patrizio. 2001. Solution Providers Unite Behind UDDI - As a directory structure, UDDI promises flexibility in B2B transactions and more muscle behind Web services. VARBusiness Manhasset. May 28th, Iss. 1711, pg. 39. Available at: ProQuest (document id: 73399013)
- [9] Wolfgang Hoschek. The Web Service Discovery Architecture. Proceedings of the 2002 ACM/IEEE conference on Supercomputing. Nov. 2002. Available at: ACM Digital Library
- [10] Win Treese. Putting it together: XML, web services, and XML. netWorker 2002. Vol. 6, Number 3. Available at: ACM Digital Library
- [11] Andrew D. Gordon, Riccardo Pucella. Validating a Web Service Security Abstraction by Typing. Preceedings of the 2002 ACM workshop on XML security. Nov. 2002. Available at: ACM Digital Library
- [12] Woodman, Morgan, Parkin. Portal replication for Web application availability via SOAP. WORDS 2003. Jan. 2003. Available at: IEEE/IEE Electronic Library (IAN 7755434)
- [13] Charles Petrie, Akhil Sahai. Business Processes on the Web. Jan. 2004. IEEE Internet Computing. Available at: IEEE/IEE Electronic Library
- [14] Henrik Frystyk Nielsen, Satish Thatte Web Services Routing Protocol (Oct. 2001) [Referenced: 25.2.2004] Available at: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-routing.asp>
- [15] Bob Atkinson, Giovanni Della-Libera, Satoshi Hada, Maryann Hondo, Phillip Hallam-Baker, Chirs Kaler, Johannes Klein, Brian LaMacchia, Paul Leach, John Manferdelli, Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Hemma Pafullchandra, John Shewchuk and Dan Simon Web Services Security (Apr. 2002) [Referenced: 25.2.2004] Available

at: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp>

- [16] Matt Migliore IMB, Microsoft and VeriSign Release SOAP Security Spec. (Nov. 2002) [Referenced: 25.2.2004] Available at: <http://www.esj.com/News/article.asp?EditorialsID=174>
- [17] Giovanni Della-Libera, Brendan Dixon, Praerit Garg, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Hiroshi Maruyama, Anthony Nadalin, Nataraj Nagaratnam, Andrew Nash, Rob Philpott, Hemma Prafullchandra, John Shewchuk, Dan Simon, Elliot Waingold, Riaz Zolfonoon. Specification: Web Services Trust Language (WS-Trust). Dec. 2002. [Referenced: 15.3.2004] Available at: <http://www-106.ibm.com/developerworks/library/ws-trust/>
- [18] Paul Roth. Predictions for software development and Web services. Computerworld. [Referenced 15.3.2004] Available at: <http://www.computerworld.com/developmenttopics/development/story/0,10801,81307p2,00.html>
- [19] IBM, Microsoft. WS-Security AppNotes. Aug. 2002. [Referenced: 20.3.2004] Available at: <http://www-106.ibm.com/developerworks/library/ws-secapp/>