ADDING SESSION AND TRANSACTION MANAGEMENT TO WEB SERVICES BY USING SIP

Wei Dong
School of Network Computing, Monash University
Unit 2/26 Morton Street, Clayton, VIC, Australia
lxbdongwei@yahoo.com

Jan Newmarch
School of Network Computing, Monash University
Monash University
Jan.newmarch@infotech.monash.edu.au

ABSTRACT

In recent years, Web services have been drawing more attention to the computer industry. They are built on open standards and provide solutions for integrating applications across enterprises. The technologies behind Web services are Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP). SOAP, which is responsible for delivering messages between applications, is a connectionless protocol. In other words, SOAP cannot keep the session state between SOAP calls. However, most E-commerce applications need to maintain the session state and transaction information between Web services. This paper proposes and develops a new mechanism that supports Web services session management by using Session Initiation Protocol (SIP) which is an IETF standard session control protocol. In addition, based on this session management mechanism, a simple transaction management solution for Web services is proposed. Two systems are implemented as demonstrations for both mechanisms in this paper. Moreover, more functions can be added in these systems, such as security, service registry and discovery.

KEYWORDS

Web services, SOAP, SIP, session management, transaction management

1. INTRODUCTION

With the development of network and distributed programming technologies, more and more software services are added in the internal network. For instance, airline companies can provide timetable services for aircrew and booking clerks. If these services could be integrated across enterprises and published on the Internet, it would bring considerable benefits to many companies. However, this requirement can still be out of the reach of dynamic Web page technologies, as different companies often use different technologies to implement their Web sites. In addition, it is difficult to accomplish application integration across enterprises without open standards. Recently, Web services which are built on open standards have been garnering much attention in the computer industry. Web services offer convenient standard ways to open up the functionality between different applications and to provide solutions when executing business transactions across enterprises.

The term Web service refers to a software system which supports the interoperable interaction between different machines over a network. It makes use of WSDL to describe its interface. Other systems can invoke Web services by using SOAP messages, which is typically transported by HTTP (Hypertext Transfer Protocol) (Booth et al. 2004). The technologies behind Web services are Universal Description, Discovery and Integration (UDDI) (UDDI 2000), Web Services Description Language (WSDL) (Christensen et al. 2001) and Simple Object Access Protocol (SOAP) (Gudgin et al. 2003). These technologies have been standardized by the World Wide Web Consortium (W3C) and generally accepted by computer industry.

The advantages of using SOAP in Web services comparing to other distributed computing techniques, such as Sun's ONC RPC (Open Network Computing Remote Procedure Call)(Ts'o and Chiu 2001), Microsoft's DCOM, CORBA's Internet Interoperable ORB Protocol (IIOP) and Java RMI are: Firstly, SOAP enables universal communication between applications and services through the Internet. Secondly, utilizing HTTP as its transport protocol, SOAP eliminates firewall barriers. CORBA/IIOP and Java RMI use object methods for RPC calls which are often denied by firewalls or proxy servers because of security problems. Finally, SOAP not only works with HTTP but also with other transport protocols such as SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) (Tsenov 2002), WAP (Wireless Application Protocol) (Liu and Chuang 2003) etc., which makes SOAP widely used in many applications and products.

However, although Web services are widely used in industry projects, some problems are arising. One is that SOAP, which is based on an XML format to transport messages between applications, is a connectionless protocol. In other words, SOAP cannot keep the session state between SOAP calls. However, in most E-commerce applications, such as a "shopping cart" or transferring money between banks, the session and transaction states need to be maintained. The traditional techniques: cookies, URL rewriting and hidden form information are not suitable to manage SOAP sessions and transactions. Consequently, new mechanisms need to be invented to manage sessions and transactions for Web services.

Currently, two main techniques are designed for adding session information to SOAP calls. One is integration of session management into Web servers. The other is adding state context in the SOAP header. However, these solutions also have their limitations which will be discussed in the next section. This paper proposes a new way of adding session information for SOAP calls by using a standard session control protocol SIP. Then based on this mechanism, it also provides a simple solution for Web service transaction management. Two Web service systems - an Online Video Shopping Web Services System and an E-banking Web Services System have been implemented as the demonstration of the proposed mechanisms.

2. BACKGROUND

2.1 Session Management

A session is defined as "Stateful connection between two parties during which one or more communications take place" (Jeckle 2002). The management of a session should include the processes of starting, joining, leaving, terminating and browsing the situations of the session across the network (Patterson, et al. 1990). There are two main issues that need to be solved in session oriented communication applications. One is how to manage multiple sessions on the server side; the other is how to keep the state between the client and server during communication.

To find a new way to manage sessions for Web services, two new session management methods on SOAP messages are explored in this section.

The first method is integration of session management into Web servers. In Tsenov's (Tsenov 2002) project named REGNET, Zap software is used to manage the session for SOAP messages. Zap is a software module which belongs to the Apache Web server. It can hide complex session management. Although, this method is easy for the developer and maintainer of the system, one disadvantage cannot be ignored. Because Zap is a part of the Apache server, it may not cooperate with other Web servers for session management. In other words, compatibility with other Web servers might be a problem for Zap. Furthermore, because the session management and the HTTP servers are bound so tightly, all the session information can be lost when the server crashes. From the programmers' point of view, the session management is like a black box. If the Web server crashes, they can do nothing to recover the session information for the Web users. If this happens, it would be detrimental for E-commerce, and would undermine Web users' confidence in E-commerce.

Another solution for adding session management on SOAP messages is by making use of the SOAP header to provide stateful services. Microsoft Corporation provides this solution for supporting the SOAP session. They define the DSML (Directory Services Markup Language) SOAP session namespace as follows:

xmlns="urn:schema-microsoft-com:activedirectory:dsmlv2"

Three SOAP header elements, <BeginSession> <Session> <EndSession>, and their specifications are also presented for operating SOAP sessions (Corporation 2003). However, this version of a SOAP session is still based on the cookie mechanism. It puts cookies in the <Session> tag, which is included in the SOAP

header (Newmarch 2004). Also, the <Session> element will be transmitted between the client and the server just like HTTP cookies. The developers have to make sure the client application treats the <Session> tags as HTTP cookies and sends them back to the server with each request. Furthermore, this mechanism assumes that SOAP is transported on HTTP which limits the cooperation between SOAP and other transport protocols.

Based on the limitations of the existing solutions, a new way of adding session information for SOAP calls is needed. Session Initiation Protocol, which is a standard session management protocol, becomes a wise choice to accomplish this goal.

2.2 Session Initiation Protocol (SIP)

SIP is developed by the Internet Engineering Task Force (IETF). It is a standard application-layer control protocol for setting up, modifying and terminating sessions regardless of media content (Cai et al. 2002). The major applications of SIP are Internet conferencing, Internet phone, Internet games and online chatting.

There are five main functions that SIP provides for operating the communication between two end users. Firstly, it can find the called user location in the communication. Secondly, SIP is able to determine the availability of the end user. Thirdly, the communication media and its parameters will be discovered by SIP and fourthly, another function sets up the session parameters for both called and calling parties. Finally, SIP can manage the session "including transfer and termination of sessions, modifying session parameters, and invoking services" (Rosenberg et al. 2002).

SIP consists of four major components: SIP User Agents, SIP Registrar Servers, SIP Proxy Servers, and SIP Redirect Servers. Firstly, SIP User Agents (UAs) can be divided into UAS (User Agent Server) and UAC (User Agent Client). Usually, these are end-user devices, such as PCs, telephones. Secondly, SIP Registrar Servers are databases which contain the UAs addresses within one domain. Thirdly, SIP Proxy Servers accept a session request from a UA, look up the callee address in SIP Registrar Servers, and then send the inviting session request directly to the callee if the callee is in the same domain. If not, the Proxy Servers will send the request to another SIP Proxy Server. Finally, SIP Redirect Servers allow the SIP proxy Server to allocate the callee address in a different domain (Rosenberg et al. 2002).

2.3 Web Services Transaction Management

The ACID transactions (serviceoriented.org) have given a clear and simple model for programmers to control transactions in a distributed environment. However, these require joined applications and components to talk to each other under a highly coupled environment (Padmanabhuni 2003). Web services are usually loosely coupled applications, and not all business applications can be strictly implemented ACID transactions. Thus, Web service based transaction management needs to be standardized.

In fact, transaction management on Web services is a new issue that is currently being addressed by industry leaders (Siegrist 2004). The Arjuna Technologies, Fujitsu Limited, IONA Technologies, Oracle Corporation, and Sun Microsystems have proposed the Web Services Composite Application Framework (WS-CAF) specification (Bunting et al. 2003). Three parts of specifications are included in WS-CAF specification: Web Services Context Service Specification (WS-CTX) (Bunting et al. 2003), Web Services Coordination Framework Specification (WS-CF) (Bunting et al. 2003), and Web Services Transaction Management Specification (WS-TXM) (Bunting et al. 2003).

The implementation of the WS-CAF consists of three steps. Firstly, a context service needs to be set up to provide context management. Secondly, a Web services coordination framework can be build on the context service to supply the message delivery function. Finally, a Web services transaction management system can be added on the Web services coordination framework to realize a variety of transaction management types.

The three Web transaction management models proposed by Bunting D. et al in the WS-TXM specification (Bunting et al. 2003) are the ACID transactions model, the long running action model, and the business process transaction model. In general, ACID transactions are still the cornerstone in these three transaction management models. However, because Web services can be applied to different kinds of businesses, in some circumstances, such as long-lived applications, the atomicity and isolation attributes cannot be strictly implemented according to the ACID transactions.

In addition, the newest movement on Web service transaction management made by W3C was the

development of Web Services Choreography Description Language (WS-CDL) (Kavantzas et al. 2004). The WS-CDL is an XML-based language which is independent of business processes, allowing an interoperable framework between different platforms and programming languages (Kavantzas et al. 2004).

To sum up, the WS-CAF and WS-CDL just give us a guideline to implement Web service transaction management. According to Siegrist J.L. (Siegrist 2004), Web services are a relatively new technology, and many issues, such as transaction management, security, interoperability etc, are needed to be considered. And the solutions of these problems are new and not complete.

3. WEB SERVICE SESSION AND TRANSACTION MANAGEMENT BY USING SIP

3.1 Web Service Session Management by Using SIP

In this section, the Online Video Shopping Web Services System is used to demonstrate how to use SIP to manage session for Web services. The system simulates a video store that sells videos through the Internet. There are six functions provided by the video store Web service and they are: *List video category, List Video, Add to Shopping Cart, Delete from Shopping Cart, Get Video Items* and *Purchase*.

As shown in Figure 1, a Web service user can establish a session with the system and then invoke Web services provided by a service provider. After the user finishes all activities with the Web services, he or she can terminate the established session.

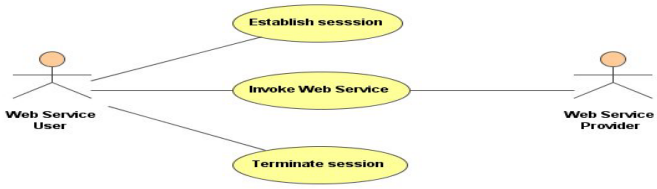

Figure 1. Use case diagram for session management on Web services

Figure 2 illustrates the architecture of the Online Video Shopping Web Services System. On the client side, there is a client application which allows users to set up a session with a SIP server and to access the video shopping Web services. The SIP server is used to manage the session between the client application and the Web services. The Web service server contains the services published by the service provider.

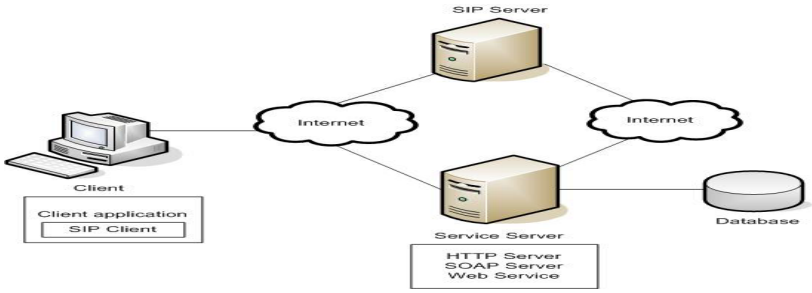

Figure 2. Architecture of the Online Video Shopping Web Services System

There are three steps involved in the proposed Web service session management mechanism. Firstly, a client and a SIP server establish a SIP session. After the session is set up, the client application has got unique session identification (ID), which is also recorded by the SIP server.

Secondly, the client and the Web service server can communicate with each other by including the SIP session ID in each SOAP messages. The session ID will be inserted into the SOAP header element and carried by every SOAP message which is transmitted between the client application and the service server. When the service server receives a user's SOAP request which contains a new session ID, it will check that session ID with the SIP server and to find out whether the client has already set up a session. If so, the service

server will first recode the session ID and then create a "shopping cart" for the client; and finally send a corresponding SOAP response message to the client.

Finally, the client needs to terminate the active SIP session after all shopping activities are finished. As a result, the session ID record is deleted from both the SIP server and the Web service server.

3.2 Web Service Transaction Management by Using SIP

The goal of this stage is to accomplish transaction management on Web services. Thus, the E-banking Web Services System focuses on the implementation of money transfer transaction management between two banks. Bank A and Bank B Web services will be implemented which provide money transferal functions. The system client can choose an account from one of these banks to transfer money to an account in another bank.

The previous stage of this paper has provided a session management mechanism for Web services, which is the first step to complete transaction management. Therefore, the mechanism will be used in the implementation of the transaction management in the E-banking Web services system. However, the session ID used in the previous stage cannot be used to identify a transaction between the two banks because a client might create many transactions within a session. In order to accomplish transaction management for Web services, a transaction manager needs to be brought in which is responsible for coordinating all the participants' activities in a transaction. Figure 3 shows the use case diagram of the system.

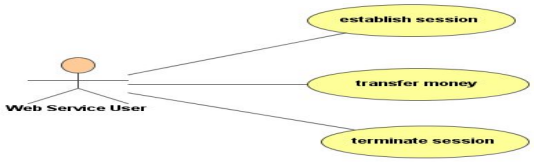

Figure 3 Use case diagram of transaction management on Web services

In this system, the Web service user can set up a session with a SIP server, transfer money between two banks (Bank A and Bank B in the system), and terminate the session with the system. The architecture of the system is illustrated in Figure 4.

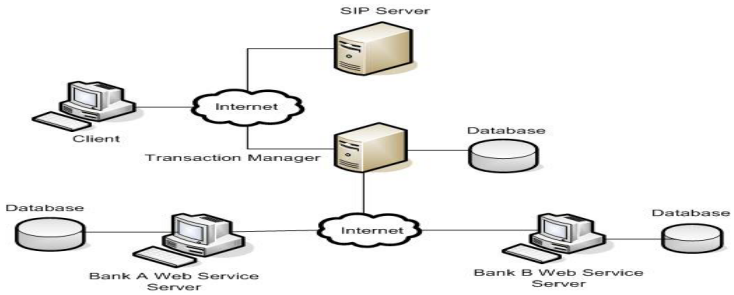

Figure 4. Architecture of the E-banking Web Services System

As with the previous system, the task of the SIP server is managing the session for all the Web services in the system. The Transaction Manager is a Web service that looks after all the transactions in the system. Bank A Web service server and Bank B Web service server are service servers that provide money transferal functions. The communication between Transaction Manager and SIP server and between Transaction Manger and the two bank Web service servers is through the Internet.

4. IMPLEMENTATION

In both of the demonstration Web service systems, the Apache Axis was used as the SOAP server. JWSDP 1.2 was used as the HTTP server. The client applications, SIP client and server and the bank-end Web service programs are developed in Java.

With the purpose of using SIP to manage the session for Web services, a SIP server needs to be running

before a user logs on the client application. The SIP server opens a server socket to accept the SIP client messages. When the Web service user starts the client application and logs onto the system, an invitation message is sent from a SIP client to the SIP server. Then the SIP server will send a 200 OK message to the SIP client. Next the SIP client sends an acknowledge message to the SIP server and the session is established between the client application and the SIP server. During the communication between the client application and the Web service server, the session ID should be included in every SOAP message. When the Web service user terminates the active session and exits the system, the SIP client first sends a BYE message to the SIP server. After the SIP server receives this message, it informs the Web service server that the user's session is terminated and then sends a 200 OK message to the SIP client. As a result, the session established between the client application and SIP server is terminated.

The session ID is the key point for accomplishing session management for Web services. Hence, generating a cyber unique identification when the session is established is the most important element for SIP implementation. In order to express the session ID in a standard manner, the "o" field of SDP (Handley and Jacobson 1998) and ISO 8601 Date/Time Representations (Kuhn) are used to generate the session ID. According to SDP specification, the format of the session ID is:

YYYYMMDDhhmmss@<SIP client IP address>

For example, a SIP client whose IP address is 192.168.0.100 sets up a session with a SIP server at 5:06:30 am, 19 December, 2004. The session ID is: 20041219050630@192.168.0.100.

In order to deliver the session ID from the SOAP message header to the back-end Web services, AXIS as the SOAP server provides an extensible message processing system for Web service programmers. By extending the basic handler class and then deploying to the AXIS engine, programmers can easily add custom functionalities. If applying this system architecture to other SOAP servers, they would have to have similar interface or function for programmers to accomplish the session information exchange.

In the Web service transaction management mechanism, the transaction manager, which is able to coordinate the transaction for all participants, is also a Web service. It conforms to the two-phase commit model which means that all participants have to agree on the transaction, only then can the transaction be committed. During the implementation of the transaction, if any participants cannot accomplish the action successfully, the participants need to rollback to the original state. A unique transaction ID is used to label every transaction created in the transaction manager. All other participants can use the transactions ID to communicate with the transaction manager, which enables data state consistency in all participants.

Although the transaction management for simple Web services has been accomplished, part of the implementation of the transaction manager would need to be changed for diverse business systems in order to coordinate different Web services.

5. SUMMARY

5.1 Session Management Mechanism

The session management mechanism proposed in this paper involves adding session information in the SOAP header element and using SIP to manage the session state for Web services. This mechanism possesses three advantages compared to similar mechanisms introduced in section 2, for example, Microsoft Corporation's SOAP session solution. These three advantages are described as follows:

- Using standard protocol to generate session ID

The session information added in the SOAP header element can vary in Microsoft Corporation's SOAP session solution. If using different ways to identify sessions on Web services, especially in E-commerce, it will be hard to integrate them across enterprises. Since Web services are based on standard protocols, such as UDDI, WSDL and SOAP, to support the interoperable interaction between different applications over a network, a standard way of describing the session ID is a suitable solution for accomplishing session management.

The advantage of the proposed mechanism is that SDP, which is developed by IETF MMUSIC for describing multimedia sessions, and ISO 8601 Date/Time Representations are used to generate the session ID. The format of the session ID is: YYYYMMDDhhmmss@<SIP client IP address>. Using this standard manner to identify a session makes it easier to integrate Web services across enterprises.

- Using standard protocol to manage the session for Web services

The Microsoft Corporation's solution for managing sessions on Web services is similar to the HTTP cookie mechanism. This means that the Web developers have to make sure that the client application treats the <BeginSession>, <Session> and <EndSession> tags as HTTP cookies and knows when and how to send those three tags to the Web server. If the Web server crashes, it will be difficult to recover the session state.

The second advantage of the current mechanism is that as SIP, a standard application-layer session control protocol, is used to manage sessions for Web services, the integration problem can be easily solved according to SIP specification. In addition, the SIP server can be located in PCs separated from the Web server. If the Web server crashes, the session state can be recovered from SIP server. Moreover, using SIP to manage the session state enhances the extensibility to Web services as more Web services can be easily added into the demonstration system.

- Extensibility on security issue

The security issue is always important for the E-commerce Web services, as customers need to input sensitive information such as credit card numbers in order to access services. However, Microsoft Corporation mentions the solution for security in their SOAP session management mechanisms.

Although in the demonstration project there is no code for implementing the security, the SIP specification can be the reference for adding the security solution to the mechanism. Actually, the SIP specification defines the communication encryption and authentication and also provides security solutions for registration, inter-domain requests, peer-to-peer requests, and denial-of-service protection. Therefore, the security issue can be easily extended in the proposed mechanism. This is the third advantage of the current mechanism.

5.2 Transaction Management Mechanism

Based on the session management mechanism, this paper has proposed a simple transaction management mechanism on Web services. In this mechanism, SIP is still used to manage the session state and a transaction manager is introduced to manage transactions among participants.

As introduced in section 2, the WS-CAF (Bunting et al., 2003) and WS-CDL (Kavantzas et al., 2004) simply provide implementation guidelines for Web services transaction management. In fact, the design and implementation of transaction management depend on the business work flow. For example, in the E-banking Web service project, the implementation of the transaction manager is based on a money transferal work flow.

The advantages of the proposed transaction mechanism are: firstly, the transaction manager is a Web service which needs to communicate with the SIP server to control transactions among joined parties. It can be changed easily according to different businesses. Secondly, as SIP is still used to manage the session state, security issue can be addressed in this mechanism according to SIP specification. Thirdly, this mechanism conforms to two-phase commit protocol which enables data state consistency in all participants. Finally, this mechanism provides great extensibility for more Web services to be added in.

6. FUTURE WORK

Instead of inventing a new protocol, this paper uses an existing standard protocol SIP to manage sessions and transactions for Web services. However, the security issue has not been considered in these mechanisms. As mentioned in section 5, SIP specification mentions various security solutions. Some of these could be added into the mechanisms in future research.

In the Web service systems, it is assumed that the service customer already knows the location of the Web service provider. Hence, the service registry and discovery procedures are omitted. In order to provide a complete Service-Oriented Architecture implementaion, a UDDI server could be added to a future system.

The transaction management is still under development because Web services are loosely-coupled, have long transaction durations and complex interactions between multiple components. This paper has provided a simple solution for transaction management on Web services. Hopefully, it will bring about new implementation possibilities for managing Web service transactions in the future.

REFERENCES

ISO 8601 Date/Time Representations. Available from http://www.mcs.vuw.ac.nz/technical/software/SGML/doc/iso8601/ISO8601.html [Accessed: 19 Dec.,2004]

Booth, D. et al., 2004. *Web Services Architecture*. Available from http://www.w3.org/TR/ws-arch/ [Accessed: Oct. 22,2004]

Bunting, D. et al., 2003. *Web Services Composite Application Framework (WS-CAF) Ver1.0*. Available from http://developers.sun.com/techtopics/webservices/wscaf/primer.pdf [Accessed: 12 Nov.,2004]

Bunting, D. et al., 2003. *Web Services Context (WS-Context) Ver1.0*. Available from http://developers.sun.com/techtopics/webservices/wscaf/wsctx.pdf [Accessed: 10 Nov.,2004]

Bunting, D. et al., 2003. *Web Services Coordination Framework (WS-CF) Ver1.0*. Available from http://developers.sun.com/techtopics/webservices/wscaf/wscf.pdf [Accessed: 10 Nov.,2004]

Bunting, D. et al., 2003. *Web Services Transaction Management (WS-TXM) Ver1.0*. Available from http://developers.sun.com/techtopics/webservices/wscaf/wstxm.pdf [Accessed: 19 Aug.,2004]

Cai, H. L. et al. 2002. *Session Initiation Protocol and Web Services for Next Generation Multimedia Applications*. Multimedia Software Engineering, 2002. Proceedings of the IEEE Fourth International Symposium on Multimedia Software Engineering pp: 70 - 80.

Christensen, E. et al., 2001. *Web Services Description Language (WSDL) 1.1*. Available from http://www.w3.org/TR/wsdl [Accessed: Oct. 22,2004]

Corporation, M., 2003. *DMSL SOAP session support*. Available from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/dsml/dsmlsoapsesssupp.asp [Accessed: 5 Oct.,2003]

Gudgin, M. et al., 2003. *SOAP Version 1.2 Part 1: Messaging Framework*. Available from http://www.w3.org/TR/2003/REC-soap12-part1-20030624/ [Accessed: Setp. 30,2004]

Handley, M. and V. Jacobson, 1998. *RFC 2327 - SDP: Session Description Protocol*. Available from http://www.faqs.org/rfcs/rfc2327.html [Accessed: 20 Oct.,2004]

Jeckle, M. C., 2002. *Adhere to Session-Oriented Communication Principles*. Available from http://www.jeckle.de/files/ssgrr2002.pdf [Accessed: 5 Oct.,2003]

Kavantzas, N. et al., 2004. *Web Services Choreography Description Language Version 1.0*. Available from http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/ [Accessed: 26 Aug.,2004]

Kuhn, M., *A Summary of the International Standard Date and Time Notation*. Available from http://www.cl.cam.ac.uk/~mgk25/iso-time.html [Accessed: 21 Dec.,2004]

Liu, C. C. and Y. D. Chuang, 2003. *Remote Data Access Scheme Support for Wireless Access to an Online Teaching System Using SOAP Technology*. IEEE Telecommunications pp: 1717 - 1722.

Newmarch, J., 2004. *A Critique of Web Services*. Available from http://jan.netcomp.monash.edu.au [Accessed: 26 Oct.,2004]

Padmanabhuni, S., 2003. *Web services transactions standards: Core requirements*. Available from http://searchwebservices.techtarget.com/originalContent/0%2C289142%2Csid26_gci913977%2C00.html [Accessed: 26 Aug.,2004]

Patterson, J. F. et al. 1990. *An Architecture for Synchronous Multi-user Applications*. Computer Supported Cooperative Work, Proceedings of the 1990 ACM conference on Computer-supported cooperative work pp: 317 - 328.

Rosenberg, J. et al., 2002. *RFC 3261 - SIP: Session Initiation Protocol*. Available from http://www.faqs.org/rfcs/rfc3261.html [Accessed: 5 Oct.,2003]

serviceoriented.org, *ACID Transactions*. Available from http://www.serviceoriented.org/acid_transactions.html [Accessed: 17 Aug.,2004]

Siegrist, J. L., 2004. *Transaction Management Issues and Recommendations for Developing Extranet - Based Web Services for the Oil Storage Company*. Available from http://www.jenikya.com/text/articles/transwebservices.pdf [Accessed: 19 Aug.,2004]

Tsenov, M. 2002. *Application of SOAP protocol in E-commerce Solution*. 2002 First International IEEE Symposium pp: 59 - 62.

Ts'o, T. and A. Chiu, 2001. *ONC Remote Procedure Call (oncrpc)*. Available from http://www.ietf.org/html.charters/oncrpc-charter.html [Accessed: 20 Oct.,2004]

UDDI, 2000. *UDDI technical white paper*. Available from http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf [Accessed: Oct. 22,2004]