SCADA systems in heterogeneous environments

A. Martinić
Power Control Systems
Končar – Power Plant and Traction Engineering Inc.
Fallerovo šetalište 22, Zagreb, Croatia
Phone: +385 1 3655 577 Fax: +385 1 3667 515 E-mail: ante.martinic@koncar-ket.hr

Abstract – Middleware communication architectures are increasingly used in modern SCADA systems. Usage of traditional middleware architectures for component integration purposes simplifies development and deployment process, but often restricts possibilities of merging different platforms. Interaction of components written in different programming languages, running on different hardware platforms and operating systems can not be easily achieved. In the last few years new XML based middleware communication architectures evolved. These architectures provide true platform and language independence, thereby making component integration in heterogeneous environments possible. In present-day SCADA systems XML technologies have not yet been extensively exploited. This article deals with SCADA data acquisition process, realized by using platform neutral XML communication architectures. Limitations of using XML middleware communication architectures in SCADA systems are identified, and the solution for dealing with these limitations is presented.

I. INTRODUCTION

Usage of middleware communication architectures in inherently complex distributed systems such as Supervisory Control and Data Acquisition (SCADA) systems is strongly advisable. Middleware architectures simplify development process and component deployment. The most important middleware communication architectures today are Distributed Component Object Model (DCOM), .NET Remoting, Common Object Request Broker Architecture (CORBA), Java Remote Method Invocation (Java RMI), XML Remote Procedure Call (XML-RPC) and Web Services. DCOM and its successor .NET Remoting can theoretically be used for interaction of components on different platforms, but in practice this proved to be difficult to achieve, thus these communication architectures are almost exclusively used on MS Windows operating systems. Java RMI requires usage of Java programming language, while CORBA demands high programming skills, and has some limitations on certain platforms. In addition, when used in networks with firewalls some of these architectures can couse numerous problems which can result with serious impact on system security. Having in mind all above mentioned disadvantages of traditional middleware communication architectures, truly platform neutral technologies XML-RPC [5] and Web Services [6] based on Extensible Markup Language (XML) seem like reasonable choice for integration of SCADA components in heterogeneous environments.

Extensive exploitation of XML based technologies started just few years ago, so there are only a small number of experimental and commercial products which utilize XML middleware communication architectures for SCADA system data exchange [1]. The major limitation of these systems emerges from the lack of notification mechanisms in XML-RPC and Web Services architectures. This mainly concerns data acquisition process conducted by SCADA systems. When employing XML communication architectures SCADA clients use polling technique to retrieve process data from SCADA servers. This means that data transfer is done periodically as initiated by client. Improvement of this technique includes subscriptions which client defines on server. Client subscribes for receiving changes regarding only specific set of process objects. When client initiates request for data transfer server replies only with the data regarding subscribed objects [10]. As a result network traffic is reduced, but main limitations of polling mechanism remain. Periodic data acquisition is not suitable for transferring data in event-based systems such as SCADA.

This paper deals with utilization of XML communication architectures in SCADA systems and proposes the solution for event-based data acquisition model implemented by employing these platform neutral architectures. SCADA system fundamentals are briefly covered in section II. The way XML middleware architectures can be utilized in data acquisition process is introduced in section III. Section IV covers the main features of the proposed model, while section V deals with security issues. Conclusions are drawn in section VI.

II. SCADA SYSTEM FUNDAMENTALS

SCADA systems are used to monitor and control technical processes that can be localized or distributed among different dislocated sites. Today SCADA systems became unavoidable part of automation systems in many processes such as power and gas generation and distribution, water supply, food production, steel industry, traffic control, etc. SCADA systems provide means for collecting data from distant locations and for transferring collected data to control centers. Different processes include different input/output points, whose numbers can range from few thousands in simple to few hundred thousands in complex systems. System control and analysis is preformed based on data collected from distant sites using different communication channels and broad range of devices, ranging from desktop computers to mobile phones. Besides real-time process monitoring, SCADA systems are used for collecting and archiving process data on which further detailed analysis can be carried out [2].

SCADA system is composed of four basic component types: SCADA servers, SCADA clients, remote terminal units (RTU), and communication equipment. SCADA system architecture is presented in figure 1. RTUs gather information from their remote sites from various input devices, like valves, pumps, alarms, meters, etc. SCADA servers collect data from RTUs. SCADA clients can be used as human-machine interface (HMI) for process visualization, data archiving, exporting data to other systems etc. Choice of communication equipment and protocols depends on the specific needs of the system.

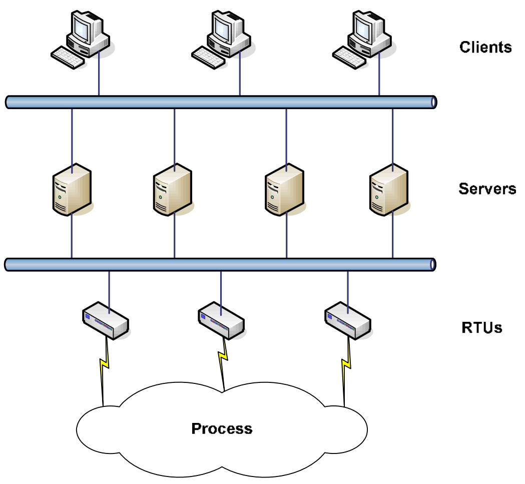

Figure 1. SCADA system architecture

SCADA server is central component which usually resides in control centre and provides means for bidirectional communication needed for data acquisition and remote device control. SCADA server initiates communication with RTUs, collects and stores process data, forwards data to other components or systems, enables process control, etc. SCADA systems define three types of data: analog, digital and state. RTUs collect process data locally and transfer colleted information to SCADA servers. Some RTUs store collected data in memory storage until SCADA server issues command for data transfer. Other, more sophisticated, RTUs use microcontrollers and programmable logic controllers (PLC) which can directly control process without intervention of SCADA server. RTU's central processing unit can interact with SCADA server using different communication protocols, whether standardized or proprietary [3].

In conventional SCADA systems client-server communication is mostly designed as event-driven communication over TCP/IP based networks. Clients apply for receiving data from process objects associated with server. When server detects change regarding desired process object it forwards received information to all clients applied for receiving information about the object in question.

III. XML DATA ACQUISITION MODEL

XML middleware communication architectures enable data exchange between components developed in different programming languages and running on different operation systems. These technologies provide higher level of abstraction than traditional operating system mechanisms like sockets, so development process is far more straightforward. XML architectures most often use Hypertext Transfer Protocol (HTTP) [9] as transport protocol. HTTP is ubiquitous protocol used to transfer textual data in heterogeneous network environments, like the Internet. The main problem of using XML architectures in SCADA systems lies in the fact that these architectures cannot provide asynchronous, event based communication. Typical examples of asynchronous events are signals and alarms, which are one of the basic data types in SCADA systems. To avoid periodical polling another interaction model must be used. This model must provide means of asynchronous data transfer from SCADA servers to SCADA clients, while utilizing platform neutral XML communication architectures. It is natural to send the information to clients not when they ask for it, but when the change producing the information really occurs.

Publish/subscribe paradigm [11] is suitable for providing described functionality. This paradigm implies existence of data sources publishing information and consumers interested in receiving information being published. Middleware architecture which implements this paradigm is commonly referred to as publish/subscribe broker. Publish/subscribe broker lies between information producers and consumers, forwarding information as it is published. XML-RPC and SOAP [7] (used as communication protocol of Web Services) are suitable for data exchange between two exactly defined components, but they do not satisfy communication including one server with many clients, or many servers with many clients. For SCADA server functionality it is not important how many clients are interested in receiving some specific information. Server even does not have to know which exactly those clients are. On the other hand, client is interested in information and not its origin (in the case when there is more than one SCADA server). Server should publish information as a consequence of changes in real world process, and client should receive all generated information regarding objects of their interest. Direct logical connection between server and client is not required, more than that it is not even desired. This kind or relationship between client and server is commonly known as loose coupling. For achieving described behavior additional XML publish/subscribe broker component must be deployed between SCADA servers as publishers and SCADA clients as consumers. Broker component implementing publish/subscribe paradigm will be utilized for dispatching collected process data among components of the SCADA system.

Architecture realizing proposed model is presented in figure 2. XML broker component does the majority of the job. It maintains information about available objects and subscriptions regarding those objects, receives data changes from SCADA servers and dispatches received data to connected SCADA clients. Process model component, while not essential for functionality of the proposed model, contributes to its generality. If technical process structure is described using process model in some standardized form it is not required by clients to locally maintain that information. That way process structure information is centrally managed and possible changes to the model do not imply modifications of client applications. Each client dynamically retrieves process structure information according to its specific needs. Received structure is interpreted and presented to the human user or to another software system. The most adequate format of storing and transferring process model information is XML notation. XML has capabilities of storing information about objects and their relations at the same time, while maintaining platform neutrality [4]. Proposed model should be as general as possible, meaning it must be applicable in different SCADA systems. That is possible because all SCADA systems share the same basic data types which will be transferred through the XML broker. Every SCADA system uses some standardized or proprietary protocol to export its data, thus it is not possible to conform all used protocols using single broker interface. The best way to maintain architecture flexibility is to use additional gateway component for each different SCADA server. Gateway components should be able to translate SCADA specific data export protocol to common XML broker interface.

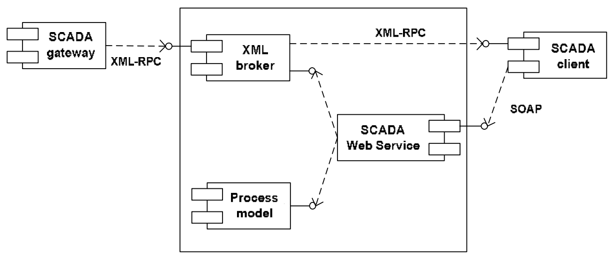

Figure 2. XML data acquisition model architecture

Different XML communication architectures are used for different purposes. Client requests (connection initialization, process model acquisition, subscription initialization, one-shot data transfer requests, etc.) are preformed through the exposed interface using Web Services. On the other hand, client notifications are preformed using XML-RPC architecture. Primary reason for XML-RPC usage lies in better performances which this architecture provides over more complex SOAP protocol used in Web Services [8]. Considerably more data is transferred from servers to clients than in the opposite direction. Having this in mind it is necessary for data transfer from SCADA servers to SCADA clients through XML publish/subscribe broker to last as short as possible so process dynamic can be adequately monitored.

XML broker exposes set of objects available through SCADA servers. Clients can explicitly, by using model or directly, choose objects whose information they want to receive. HMI clients have additional possibility of implicit subscription activation. Process model should contain information about different schemes which visualize different parts of the process. When human operator chooses scheme to be displayed all subscriptions regarding objects presented on the scheme should be automatically initiated. This implicit subscription activation uses object relations described in the process model and relieves user of one by one subscription activation. Process data regarding objects of interest are then delivered to SCADA clients using notification calls realized through the XML publish/subscribe broker. As a consequence, need for server polling and periodic information transfer is eliminated. Basic component interaction in the case of implicit subscription activation is presented in figure 3.

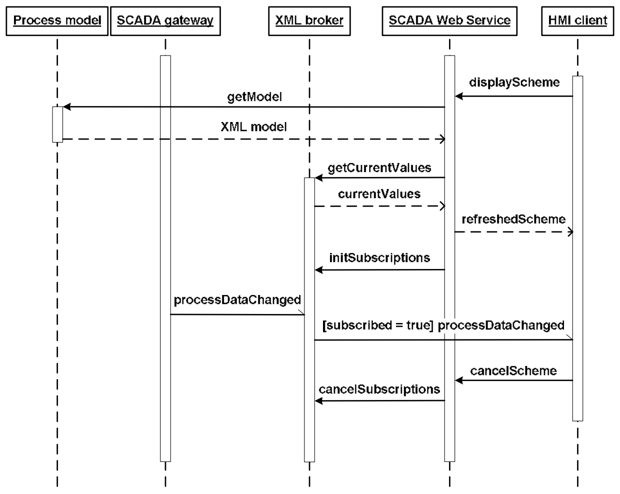

Figure 3. Basic component interaction

Described architecture provides flexibility and as such it can be easily extended. One example is addition of E-mail or SMS notifications. These notifications are complementary to the ones previously described, because they are designed as messages intended for human operators. Typical messages that can be sent this way are alarms which signalize some critical process situations. This fits into the proposed model in a way that an additional client acting as E-mail or SMS gateway has to be deployed and properly configured. This client then can receive information through XML broker notifications, compose proper messages and send them to all interested parties.

XML data acquisition model allows exportation of SCADA process data to different client systems in heterogeneous environments. By utilization of the described architecture one gets functionality of the conventional SCADA systems combined with platform neutrality provided by XML technologies. Until now SCADA system communication in heterogeneous environments implied using low level mechanisms like sockets as a mean of data exchange between components on different platforms. By using XML middleware technologies like XML-RPC and SOAP developer deals with higher level of abstraction. Some other platforms, beside standard desktop operator stations in private corporate networks, can be used. Public network such as Internet can be utilized. This enables mobile computers with wireless Internet connection, PDAs and mobile phones with XML support to be used in monitoring process. Utilization of XML technologies enable SCADA systems to be easily deployed in truly heterogeneous environments.

IV. XML PUBLISH/SUBSCRIBE BROKER

A. XML broker architecture

To be able to simultaneously distribute data to all connected clients it is essential for XML publish/subscribe broker to be designed as a multithreaded process. Every client has dedicated thread that performs its notifications. Each new client initiated connection causes a new notification thread to be created within XML broker process. Each thread must include a buffer for temporary data storage due to different throughput and possibility of communication congestions towards clients residing in different parts of the network. This architecture characteristic is especially important when many system changes happen in a short time and as a result large quantities of information have to be dispatched. In conventional SCADA systems deployed in private corporate networks this problem is not frequently manifested for several reasons. One reason is relatively high and stable throughput towards all installed clients. Number of clients is usually limited to few clients installed on desktop stations with similar performances. Network communication is fast and used hardware is powerful so large amounts of data can be transferred and processed in short time interval, thus there is no need for additional buffering. When communication takes place over heterogeneous networks, especially Internet, many different platforms with different processing capabilities can be involved. Variety of devices can be used, from classical desktop computers with different hardware capabilities, mobile computers, PDAs and even mobile phones. When throughput of the network separating data source and destination is also taken into consideration it is clear that different clients on different locations can receive data with significantly different dynamic. This is especially important when using mobile devices that can be temporary disconnected from the network. Temporary disconnections result in temporary incapability of data receiving and processing [12]. When independent data transfer towards each client would not be used, transfer dynamic would be dictated by the client with the lowest performances. This would impose serious restrictions which would, almost certainly, lead to complete uselessness in practical appliances.

Each new publishing thread created by client uses SCADA server as its data source. Components of the proposed architecture, including XML broker, must be platform neutral so they can conform requirements imposed by heterogeneous environments maximizing possible field of usage. XML broker has to receive process data through platform neutral XML-RPC architecture. This way it is possible for broad range of SCADA servers to dispatch its data using XML broker component. Existing SCADA systems are rarely capable to export real-time data in XML format, so additional gateway components must be used. Gateway components provide a way for different SCADA systems to export its data without the need for any modifications of the XML broker. Once the whole model is implemented all that is needed to connect new SCADA server is to create and deploy adequate gateway component. There are some industrial standard protocols (e.g. OPC) that enable standardized way of data exchange in SCADA systems. When gateways translating these standard protocols to XML interface calls are implemented it is possible to use XML data acquisition model in all systems utilizing protocols in question.

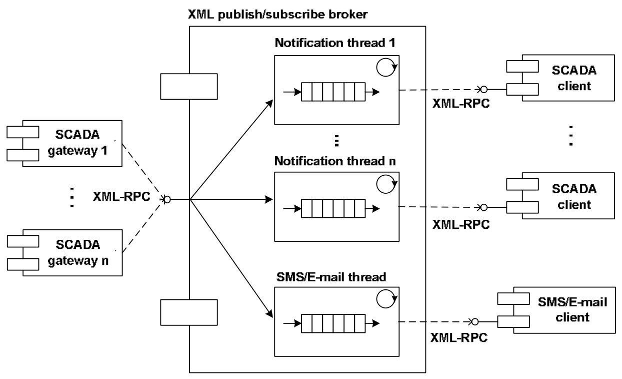

Figure 4. XML broker architecture

SCADA clients as data consumers receive notifications using XML-RPC middleware communication architecture. To be able to receive data each client must implement listener object. Object listener used by each client acts as XML-RPC server receiving data from associated XML broker notification thread. Received data is forwarded further to the business layer of the client application.

SCADA server communicates solely with XML broker through related gateway component, while XML broker component is responsible for further data dispatching. This way SCADA server, being the real data source, bears the burden of only one extra client – gateway component. For SCADA server it is completely irrelevant how many clients are connected to the system. Server does not know how many clients are there and does not contain any references to connected clients. Burden is transferred from SCADA server to XML broker component, thus minimizing impact on employed server. This is one of the most important characteristics of the proposed architecture. In practical appliances it is very important that deployment and utilization of the architecture supporting heterogeneous communication has the smallest possible impact on existing SCADA system. On the other hand, all that is needed by the clients is knowledge about exposed interface. Clients can retrieve process schemes, request data and initialize notifications while not knowing which SCADA server supply requested information. It is irrelevant whether there is only one or there are many SCADA servers publishing over XML broker component. Number of server components depends of the nature of the system and can be submitted to changes without any influence on client application structure.

B. *XML broker subscription model*

Subscription model describes the way that clients utilize to activate subscriptions regarding objects representing technical process elements. The simplest way of subscription activation is explicit activation regarding each and every object of interest. If SCADA client is used to forward collected data to other software systems and does not provide some sort of process visualization intended for human interaction explicit activation can be acceptable mean of subscription initialization. The main reason is that for this kind of clients set of the data required is usually known in advance and it is rarely modified, so maintaining object list is not a demanding task. If frequent and dynamic change of subscriptions is needed than explicit subscription activation is not quite practical. The finest example of this kind of client are HMI clients where every change of displayed scheme means canceling subscriptions to objects shown on the previous scheme and activating subscription regarding objects presented in new scheme. To make subscription activation simpler from the user point of view, there is a need of using subscription categorization meaning that clients can on a one-time basis initialize subscriptions regarding defined groups of objects. To make this kind of subscription categorization possible process model that describes object relations must be present.

Combined subscription model, consisted of topic-based and content-based subscription paradigm, is convenient for appliance in XML data acquisition model. Topic-based model is suitable for activating subscriptions according defined object categories, while content-based model proves to be useful for making selections in particular group, according to defined criterion [11]. Topic-based model is suitable for dividing process objects into logical groups. For example, process objects can be grouped according to process schemes which are presented to the operators via HMI clients. Process schemes are usually organized hierarchically starting from one global view representation of the whole process to the views that represent only a small group of related objects, even a single object. When human operator selects desired picture subscriptions to all objects presented on the picture should be automatically activated. One section of the observed process is associated with one topic of the topic model. Subscription to the topic means subscription activation on all objects belonging to the logical group of objects represented by selected topic. Content-based part of subscription model is useful for filtering information within specified topic. This way XML broker can, on behalf of the client, filter messages from specified object group according to defined criteria. Criteria used for content-based selection can be arbitrarily complex.

Message categorization against defined topics is relatively simple procedure for whose implementation lookup tables can be used. Lookup tables are used for determining which clients must be notified when certain information is changed. This procedure is not complex or processor demanding operation. On the other hand, content-based selection can be quite complex and significantly increase processor load, depending on how complex selection conditions are. Therefore, it is necessary to make some compromises when specifying selection conditions, having in mind desired performances of the system. Subscription model with according lookup algorithms directly affects performances, so it is one of the key factors in the proposed XML acquisition architecture. Although combined subscription model imposes more load to the system than solely topic model, its use is desirable mainly because it is less effective to needlessly send information to clients than to filter information according to well defined criteria.

Presented subscription model optimizes network traffic and reduces processor load. Significance of subscription model usage increases as observed process becomes more dynamic. Present-day SCADA systems are generally designed to conform high throughput private networks, so most often clients, especially HMI clients, are refreshed with the whole set of process data available. If operator is not interested in actively observing all available objects, some unnecessary information, which causes increased system load, is also transferred. This does not cause any significant problems in high bandwidth networks, but systems using low bandwidth networks and less powerful hardware can experience substantial congestions, thus network and processor usage optimization is of great significance.

V. SECURITY ISSUES

If data transfer takes place over insecure networks information security becomes important and unavoidable system feature. Process data set, or at least some of its subsets, is very often treated as a business secret, hence transferred data must not be available to unauthorized parties. Information in transfer can be compromised by unauthorized reading or by influencing information content. The whole traffic regarding security sensitive information must be protected from malicious attacks using adequate security techniques that provide party authentication, data confidentiality and data integrity [13].

XML-RPC and Web Services architectures are used to conduct all data exchange in the XML data acquisition architecture. XML-RPC and SOAP messages are XML documents, which in turn are more or less simple text documents. HTTP is ubiquitous protocol for transferring text documents in heterogeneous environments and as such it is most often used as transport protocol in XML communication architectures. HTTP communication is in most cases secured by utilizing Secure Socket Layer (SSL) protocol. SSL became, with some slight differences, IETF standard known as Transport Layer Security (TLS) [14]. HTTP over SSL/TLS is commonly referred to as HTTPS [15]. SSL/TLS manages entity authentication and secure data transfer, and as such can be used for securing all communication in XML data acquisition model. Authentication in SSL/TLS is assured by digital certificate exchange, while transferred data is shielded by using combination of symmetric and asymmetric cryptography. Asymmetric cryptography is used during connection establishment process, while more efficient symmetric cryptography is used in actual data transfer [16].

In the proposed model three communication directions can be identified: data export from SCADA server to XML broker, client requests to XML broker and procedure of data dispatching from XML broker towards connected clients. SCADA server gateway and XML broker employ unidirectional communication in which information is sent from SCADA server, over related gateway, to XML broker. Security settings regarding this communication depends on XML broker location. If connection between SCADA server and XML broker must be established over insecure network, necessity to use cryptography methods emerges. It is necessary to perform authentication of the XML broker before any data is published from SCADA server. This assures that process data is sent to the desired destination. SCADA server authentication is also required to prevent malicious impersonations of the server that would result with distribution of data which does not originate from the process actually monitored. All data sent from SCADA server to XML broker must be encrypted so man-in-the-middle attacks, like unauthorized reading and content alternation, can be prevented.

XML broker and connected SCADA clients employ two-way communication. Interaction is initiated by the client, and mutual authentication must be preformed. XML broker should be authenticated by digital certificate, while client can prove its identity either by digital certificate or by using some other means like providing username and password. When authenticated client can retrieve available information about the system, initialize and activate subscriptions, etc. Communication from XML broker to a client is analogous to SCADA server gateway to XML broker communication. The only difference is that XML broker can interact with many clients, but this does not require any additional logic, hence described security techniques are simply applied separately for each client.

VI. CONCLUSION

In present-day SCADA systems utilization of XML technologies is quite limited. XML is mostly used to store and exchange static information about observed technical processes. Intention of this paper was to present new possibilities of utilizing XML technologies in SCADA systems. XML data acquisition model presented throughout this paper offers functionality of conventional SCADA systems while using XML based communication architectures. Major advantages provided by XML middleware architectures include platform neutrality, simplicity of development and flexibility of deployment. Broad range of XML enabled programming languages, operating systems and hardware platforms can be combined to implement and run components in distributed SCADA system. Possibility of using mobile computers, XML enabled PDAs and mobile phones in conjunction with public Internet network extend and enable new ways of process monitoring. By using XML architectures in dynamic SCADA processes possibility for full exploitation of benefits of distributed heterogeneous environments becomes reality.

REFERENCES

[1] M.Žagar, D.Antonić, M.Orlić, D.Fudurić, T.Sečen, "Analysis of SCADA systems for power generation and distribution monitoring", Zagreb, 2004. (in Croatian)

[2] A.Daneels, W.Salter, "What is SCADA?", *International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, 1999. http://epaper.kek.jp/ica99/papers/mc1i01.pdf

[3] Synchrony Inc., "Trends in SCADA for Automated Water Systems", White Paper, 2001. http://www.synchrony.com/trends_SCADA.pdf

[4] A.deVos, S.E.Widergren, J.Zhu, "XML for CIM model exchange", *22nd IEEE Power Engineering Society International Conference*, p.31, 2001. http://www.langdale.com.au/PICA/CIMXML.pdf

[5] UserLand Software Inc., "XML-RPC Specification", 1999. http://www.xmlrpc.com

[6] D. Booth, H.Haas, F.McCabe, E.Newcomer and others "Web Services Architecture", W3C, 2004. http://www.w3.org/TR/ws-arch/

[7] N.Mitra, "SOAP Version 1.2 Part 0: Primer", W3C, 2003. http://www.w3.org/TR/soap12-part0/

[8] M.Olson, U.Ogbuji, "Choose the best tool for the task at hand", IBM library, 2002. ftp://www6.software.ibm.com/software/developer/library/ws-pyth9.pdf

[9] R.Fielding, J.Gettys, J.C.Mogul, H.Frystyk and others, "Hypertext Transfer Protocol – HTTP/1.1", RFC 2616, 1999. http://rfc.net/rfc2616.html

[10] OPC Foundation, "OPC XML-DA Specification", 2003. http://www.iconics.com/support/PDFs/ OPC_Specs/OPC%20XMLDA%20Specification.pdf

[11] P.Th.Eugster, P.Felber, R.Guerraoui, A.M.Kermarrec, "The Many Faces of Publish/Subscribe", *ACM Computing Surveys*, vol. 35, p. 114, 2003. http://www.eurecom.fr/~felber/publications/CS-03.pdf

[12] Y.Huang, H.Garcia-Molina, "Publish/Subscribe in a Mobile Environment", *2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, 2001. http://www.db.stanford.edu/~yhuang/papers/mobpubsub.pdf

[13] C.Wang, A.Carzaniga, D.Evans, A.L.Wolf, "Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems", *International Conference on System Sciences*, 2002. http://www.cs.virginia.edu/~evans/pubs/hicss.pdf

[14] Zeus Technology, "SSL: Theory and Practice", Cambridge, England, 2000. http://support.zeus.com/doc/tech/ssl.pdf

[15] K.Naqvi, "Working of HTTPS (HTTP over SSL/TLS)", Infosys Technologies, 2002. http://www.infy.com/Technology/Working-of-HTTPS.pdf

[16] PGP Corporation, "An Introduction to Cryptography", California, USA, 2004. http://download.pgp.com/pdfs/Intro_to_Crypto_040600_F.pdf