Efficient Web Service Based Data Exchange for Control and Monitoring Systems

Vu Van Tan, Dae-Seung Yoo, and Myeong-Jae Yi

School of Computer Engineering and Information Technology,
University of Ulsan, San-29, Moogu-2 Dong, Namgu, Ulsan 680-749, Republic of Korea

E-mail: {vvtan, ooseyds, ymj}@mail.ulsan.ac.kr

Abstract

Web application-based OPC (Openness, Productivity, and Connectivity) technique to exchange data between the measurement and control systems on the plant floor with XML leverages is proposed in the OPC XML-DA (Data Access) Specification using for slow control systems. In addition to the automation and control systems, the high performance and ability to exchange the complex data between the collaborating applications and applications based on non-Microsoft Platforms are widely required for designing and deploying. This paper proposes the design and implementation of Web Service for control and monitoring systems. Our Web Service is integrated services for accessing real-time and history data from control networks. This system allows the OPC XML-DA Clients to read and decode any type of data from measurement and control systems on the plant floor. Our system model fully applies such technologies as OPC, XML, and links to the Internet. Moreover, the security solutions are carefully discussed for choosing an optimal solution to apply in our system. In addition, the real-time performances of our system are considerately analyzed. The experimental results and performance evaluations have obviously demonstrated that our design and implementation have an acceptable performance and are feasible for high speed data exchange in DCS (Distributed Control System) today.

Keywords: OPC, complex data, Web service, XML, control systems, real-time performance, high speed, DCS, security

I. Introduction

Web technologies are gaining increased importance in automation and control systems, especially in slow control systems. In addition, XML is an open standard that provides interoperability and data integration using the Internet. It also is the preferred format for encoding and moving the structured data in the independent system. XML and Internet technologies provide new and powerful ways of assessing and delivering the plant floor, condition monitoring, e-diagnostic, etc. to users across manufacturing enterprise using standard Web-browsers and wireless devices [1]. Therefore, the OPC Foundation formed the OPC XML-DA technical working group to define a new specification to move the same type of plant floor data as the existing OPC COM-DA. The OPC XML-DA provides vertical integration between the plant floor and condition monitoring system, maintenance system, and enterprise application using such as industrial standards, XML, and SOAP (Simple Object Access Protocol) [2], [3]. Additionally, the OPC Foundation has defined the OPC Complex Data for implementing both the OPC DA and XML-DA. The OPC Complex Data working group is making enhancements to OPC Specifications based on requirements and feedback from other industry groups to address additional data types such as structures, binary, XML, etc. [4]. The design and implementation of the XML-DA Server that allows the OPC Clients to read and decode any type of data from the hardware I/O devices (i.e., measurement and control systems) on the plant floor are represented in [2]. Another aspect mentioned in [5], the authors proposed a data exchange protocol based on the OPC XML-DA. This protocol has focused on the acceptable performance by providing several extensions. It was interested in the consistency with high level standards, multi-platform compatibility, and high performance.

However, the requirement to convert data between the XML representation and OPC binary representation make it to determine the required size of the memory buffer to hold the whole data. Moreover, the only fast possibility to transmit large amount of data is to use binary representation rather than XML representation. In addition, different platforms even different compilers of the same platform use different binary representations such as different floating point formats, different string formats, etc. [4], [6]. Therefore, a few universal standards should be chosen to transport data between different platforms. Besides these problems, the OPC XML-DA has a big disadvantage that uses XML textual data representation. This causes much more network traffic to transfer data [6].

This paper proposes the design and implementation of Web Service for automation and control systems that are based on the OPC and XML techniques. These design and implementation include interfaces, functionalities, and architectures. Our model can issue static and dynamic web pages as well as provide interface for clients to control and monitor the field devices. By using the binary data representation, our Web Service provides a good performance for industrial automation systems. In addition, our system allows the OPC Clients to read and decode any type of data from the hardware I/O devices. In addition, the security solutions are carefully discussed and analyzed, and the optimal security solution based on XML Distributed digital signature is applied in our system.

This paper is organized as the following sections: The next section shows a model of Web integration, overviews of related OPC techniques, and the problem statements. Section III proposes a framework for design and implementation of the Web Service. Section IV investigates the real-time data publishing, data connection and data representation between Web Server and OPC Clients with the results of the benchmarks. The purposes of them are to guarantee the high performance of our Web Service. In Section V, the security aspects are carefully discussed and then the optimal security solution is exposed for applying in our system. The experimental results and evaluaions of real-time performances are appropriately estimated and represented in Section VI. They indicate that our system has an acceptable performance and is feasible to apply for control and monitoring systems in industry today. Finally, we mark some conclusions and future work in Section VII.

II. Background and Problem Statements

We show the model of Web integration in Section II-A. The overviews of related OPC techniques, which are used in automation and control systems, are represented in Section II-B. Finally, the problem statements are adequately discussed in Section II-C.

A. Model of Web Integration

A general architecture of Web integration consisting of three layers is shown in Fig. 1 [7]. The lower layer provides information from automation devices to controller level of an automation system. The upper layer is based on standard IT technologies such as Client-Server model, using Web Server as data source and Web browsers as clients. The middle layer contains the functionalities of business logic and performs as an application gateway between the upper clients and lower automation systems. The Web Server can be used to assign information from the automation and control systems to object models that can be accessed via COM/DCOM (Distributed Component Object Model). The most important problem is the clear data mapping between data and Web application because the data have different types and different semantic

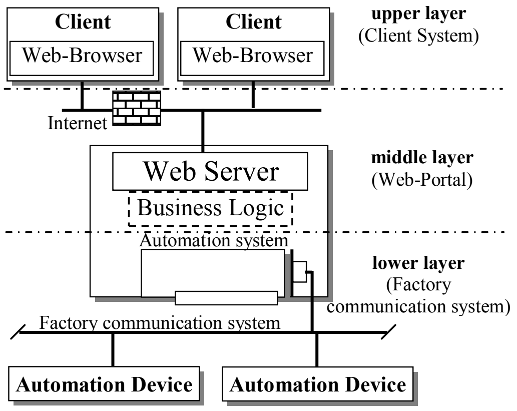

Fig. 1. Generic architecture of Web integration for process monitoring systems

meaning. Using Web technologies for slow control systems, it means integrating a multitude of different technologies. Based on this model, the XML-DA Services are implemented in the middle layer to provide the Web Services for the clients to invoke.

B. Overviews of Related OPC Techniques

The OPC Foundation, formed in 1996, is independence, non-profit, industry trade association comprised of more than 300 leading automation suppliers worldwide. This section provides some overviews of related OPC technologies. Developed by an automation software and hardware consortium, OPC is the first automation-domain-specific component standard. OPC standardizes the mechanism for communicating to numerous data sources, whether they are devices on the plant floor or databases in control rooms. OPC Interfaces provided by the server let any client access the server devices. In OPC Server-Client model, server applications acquire, contain, and serve data to the client applications. OPC Servers provide a standard interface to OPC Objects, letting OPC Clients applications exchange data and control commands in a generic way. Each OPC Client can communicate with one or more OPC Servers from different suppliers. OPC Client applications access data in the same way, whether the data are coming from an OPC Server connected to a PLC (Programmable Logic Control), industrial networks like Foundation Fieldbus, Profibus, DeviceNet; or SCADA (Supervisory Control and Data Acquisition system), or a laboratory information management system, or a production management system, etc.

However, the data from the plant floor need to significantly exchange between applications running on different platforms. Therefore, the OPC XML-DA [3] defines a new specification to move the same type of plant floor data as existing OPC COM-DA products. This provides vertical integration between the plant floor and condition, monitoring, maintenance, etc. using XML, HTTP, and SOAP industry standards. The OPC XML-DA provides better connectivity and interoperability for production management and enterprise applications such as MES, ERP, CMMS, EAM, and plant optimization that need to access to the plant floor data. In addition, it is complementary with products based on the existing OPC DA Specification [8]. The OPC XML-DA was specifically designed to allow the existing OPC COM-DA products to be wrapped by the OPC XML-DA Interfaces and in effect support both interfaces from the same OPC Server. It is as a standard Web Service Interface for reading and writing data from and to plant floor automation systems [3]. Any group can develop a generic OPC XML-DA Wrapper to Internet-enable existing OPC DA Servers allow them to publish the plant floor data to the Web. The OPC DA and OPC XML-DA provide the plant floor to manufacturing enterprise integration as shown in Fig. 2. OPC XML-DA Data model is based on OPC Items that are named and organized in a hierarchy. Each OPC Item stores a single value and is defined with

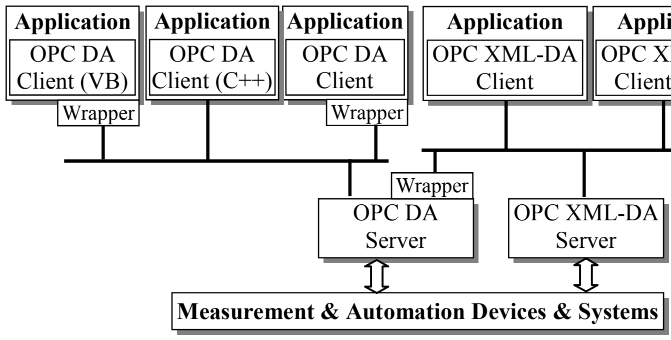

Fig. 2. The OPC DA and OPC XML-DA provide plant floor to manufacturing enterprise integration

combination of two strings: *item path* and *item name*, the *item path* identifying a namespace in which the *item name* is unique. A set of dynamically retrievable properties is associated with every item containing its metadata including human readable description, access rights, a timestamp, change rate, data type. Operations for accessing item values are *Read* and *Write*. Both operations allow accessing several items with single call. To optimize periodic reads of the same set of items as subscription mechanism is provided. The set of items is subscribed by calling the operation *Subscribe* and then periodically polled using *SubscriptionPolledRefresh*. In addition, the operation *Browse* and *GetProperties* are used to query which OPC Items are available and value of their properties. *Browse* allows querying an OPC Item's immediate successors including filtering and can return property values of the item found. The operation *GetStatus* is used to retrieved the status of OPC Server. Finally, the OPC Complex data initiative will provide a full way for the OPC Clients to read and decode any type of data from the hardware I/O devices on the plant floor. Complex data mean that an OPC Item is defined as a structure. The item includes read-only information, run-time status, and writeable control points. Actually, the complex data contain complex data items that can include non-structured items, structured items, XML data, OPC Binary, integer, etc. [4]. The OPC Complex Data specification defined two type systems that provide this level of capability, *XML Schema* and *OPC Binary*. The XML Schema describes complex data values that represented in XML, the OPC Binary that describes complex binary values defines the format of binary dictionaries.

C. Problem Statements

As a big step for process control, the OPC Foundation has created a new XML-DA to allow wrapping existent OPC DA that is widely successful standard based on COM/DCOM. Thus, PLC, DCS, HMI (Human Machine Interface) and other factory-floor software vendors use the OPC standards to compatibility and interoperability. Recently, there are several proposals have been focused on the Web Services for control and monitoring systems in industry as mentioned in [2], [5], [6], [7], [12], [33]. The protocol mentioned in [5] was interested in the consistency with high level standards, multi-platform compatibility, and high performance systems. However, the performance of this system has not provided yet. Other proposal presented the design of Web Service based Embedded Web Server [12] to propose architecture of embedded system for control systems based on OPC XML-DA. But, the performance evaluation related to architecture design has not exposed. A research on OPC XML-DA is represented in [33], the authors have developed a prototype embedded XML-DA Server on their controller, which sends or receives information instead of the field devices and the clients communicate with the controller. The performance evaluation was provided to indicate that their system has an acceptable performance. As an industrial standard to transfer data from the plant floor to the enterprise systems, the OPC XML-DA solution provides more advantages that have been represented in Section II-B. However, the OPC XML-DA solution has a big disadvantage that is to use the XML data representation [3], [6]. The XML data representation causes much more network traffic to transfer data. Moreover, data alignments are often required transport data in native representation. The OPC Complex data based on using XML presentation requires big amount of memory and high intensity of memory management operations. In addition more CPU resources for transformation between native data representation and XML representation are required [9]. A further issue is the present unavailability of XML versions of OPC HDA (Historical Data Access) Specification [10] and OPC AE (Alarms and Events) Specification [11] that are typically required in scientific experiments. In [2], the authors provided the design and implementation of OPC XML-DA Server that allows the OPC Clients to read and decode any type of data from measurements and control systems. However, they have not provided any the evaluation of performances. Additionally, the OPC XML-DA requires XML messages to be very descriptive about the data being transferred. For example, instead of "0.555" the record *<value xsi:type = "xsd:float">0.555</value>* is sent, that means the bandwidth is increased about 6 times or even more [6]. Besides these problems, the requirement to convert the data between the XML representation and OPC binary representation make it to determine the required size of the memory buffer to hold the whole data. In addition, the only fast possibility to transmit large amount of data is to use binary representation rather than XML representation. However, different platforms even different compilers of the same platform use different binary representation such as different floating formats, different string formats, etc. To resolve these problems, we propose an efficient Web Service based on OPC XML-DA to transfer data from the hardware I/O devices on the plant floor to the enterprise application level with its acceptable performance.

III. Design and Implementation

In this section, we represent a framework for design and implementation of Web Service. This Web Service allows the OPC XML-DA Clients to read and decode any type of data from the OPC Servers or measurement and control systems (i.e., hardware I/O devices). The system aspect design and the module design are represented in Section III-A and Section III-B, respectively. The Web Services based on OPC XML-DA to illustrate our work are significantly exposed in Section III-C. Finally, the XML based description model for Web Service is considerately discussed in Section III-D.

A. The Design of System Aspects

Web Server runs in multi-task (i.e., multi process or/and multi thread) preemptive system and it gives a firm support for all HTTP versions. In order to design and implement efficient Web Service for supporting several kinds of the OPC Clients to read and decode any type of data from the hardware I/O devices and high speed data exchange in DCS, we represent the aspects of Web Service implementations as shown in Fig. 3. These aspects indicate that our system supports not only for browser-based clients, but also for application-based clients. According to the OPC XML-DA Specification, our Web Service is implemented services in both the *SOAP Request Handler* and *Service Aggregator*. The Service Aggregator is integrated the services, which are based on the OPC XML-DA, in *OPC XML-DA* module. As we mentioned, the binary data representation is more effective and optimal than the XML textual representation. Therefore, our implementation should consist of several modules for supporting the OPC Complex data and improving the limitation of performances of the OPC XML-DA based on the binary data representation. The SOAP request handler will listen to the services that are invoked by the clients and the appropriate results are returned. These services are generated from the corresponding WSDL (Web Service Description Language) [13].

Communication between the OPC Clients and Web Service should provide some mechanisms to monitor and control the hardware I/O devices, sending web pages to clients and dealing with the submissions of the clients. This Web Service should also provide security

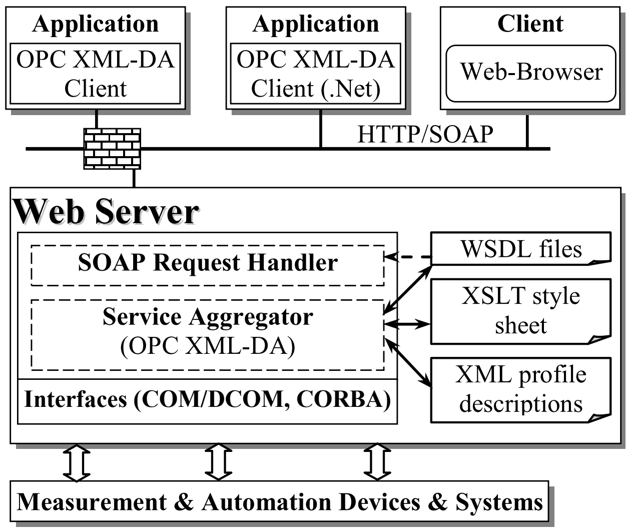

Fig. 3. The aspects of Web Service implementations

interface for remote clients. Modules in Web Service are set to respond clients' SOAP requests. These modules parse the SOAP messages received, handle the contained requests, access the corresponding real-time or historical data, and format them as the XML documents. These modules also perform the data transformation, data encryption/decryption, and configurations. In addition, the OPC Complex data can be described in WSDL file. Actually, this file is XML files that contain such service information on data types, value ranges, display information, parameters, etc. To reduce a number of transformations of the temporary XML files, the XSLT (eXtensible Style Language for Transformations) [14] is used in our Web Service. Finally, these modules send an XML document with the output format back to the OPC Clients as a SOAP response. The communication between the Web Service and OPC Clients is generically shown as in Fig. 4.

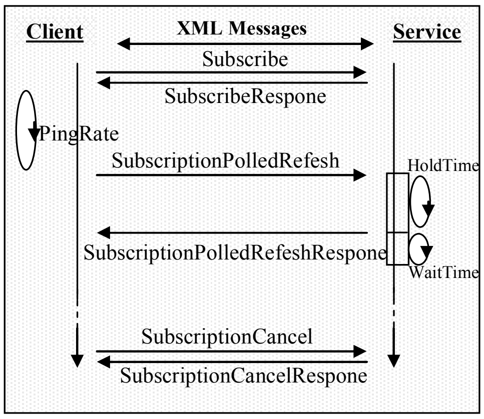

Fig. 4. The communication between the Web Service and OPC Clients

B. The Design of Modules

In order to design and implement our Web Service, we firstly present a module, *OPCXML-DA* module, which is used to manage the complex data, server, and address space (e.g., the OPC Groups and OPC Items, etc.), is based on the OPC XML-DA Specifications [3]. The *OPCXML-DA* module is fully aggregated by three classes such as *DataManagement*, *ServerManagement*, and *AddressSpaceManagement* as shown in Fig. 5. Firstly, the *DataManagement* class manages a data buffer. It consists of some functions to refresh data value and timestamp of OPC Items and to check availability of data. This class uses the

OPCComplexData class that provides the methods to represent and convert the complex data from the hardware I/O devices that complex data types are defined as dictionaries in WSDL file. Hence, each OPC Item is defined as an OPC Complex data item. The *ServerManagement* class contains several functions to listen to new connections, hold old connections, manage a number of clients, etc. It initializes field bus information, gets all parameters, and listens to the requests from the clients. When a request comes, it will create an OPC Server and responds to the OPC Clients all kinds of requests such as browse address space, add/delete Groups, add/delete Items, read/write, data subscription, data refresh, etc. are executed by *AddressSpace-Management* class. Finally, the *AddressSpaceManagement* class performs basic functions of the OPC XML-DA such as *add/delete Groups*, *add/delete Items*, *read/write*, *subscription*, etc. Besides, the *OPCXML-DA* module also uses the *XMLDA-Wrapper* module that is used for wrapping the OPC DA products (e.g., all OPC DA versions). This *XMLDA-Wrapper* module aggregates three classes, *Subscription*, *RemoteSubscription*, and *Server* as shown in Fig. 5. Basing on the OPC XML-DA Specification, "*Polled-pull*" model is adopted between the OPC Clients and the OPC XML-DA Server. This means that when an OPC Client sends a subscription to a certain items and the OPC XML-DA Server receives this request, the server does not respond immediately. After receiving the OPC Client's *SubscriptionPolledRefresh*, the OPC XML-DA Server gives changed items to this client. In our Web Service, the contact relation between the client and server can be optimized by using three supplementary attributes to *Subscribe* method such as *RequestedSamplingRate*, *EnableBuffering*, and *Deadband*.

As shown in Fig. 3, the *SOAP Request Handler* receives the SOAP requests from the clients, looks up the appropriate service in the implementation class, invokes the request methods, and returns the requested results as the SOAP responses to the clients. Therefore, we have implemented the *XMLDA-Service* class in the *SOAP Request Handler* as shown in Fig. 6. This *XMLDA-Service* class normally provides several functions such as *GetStatus*, *Subscribe*, *SubscriptionPolledRefresh*, *SubscriptionCancel*, *Read*, *Write*, *Browse*, etc. These functions are used for processing the SOAP requests and return the results corresponding to SOAP requests

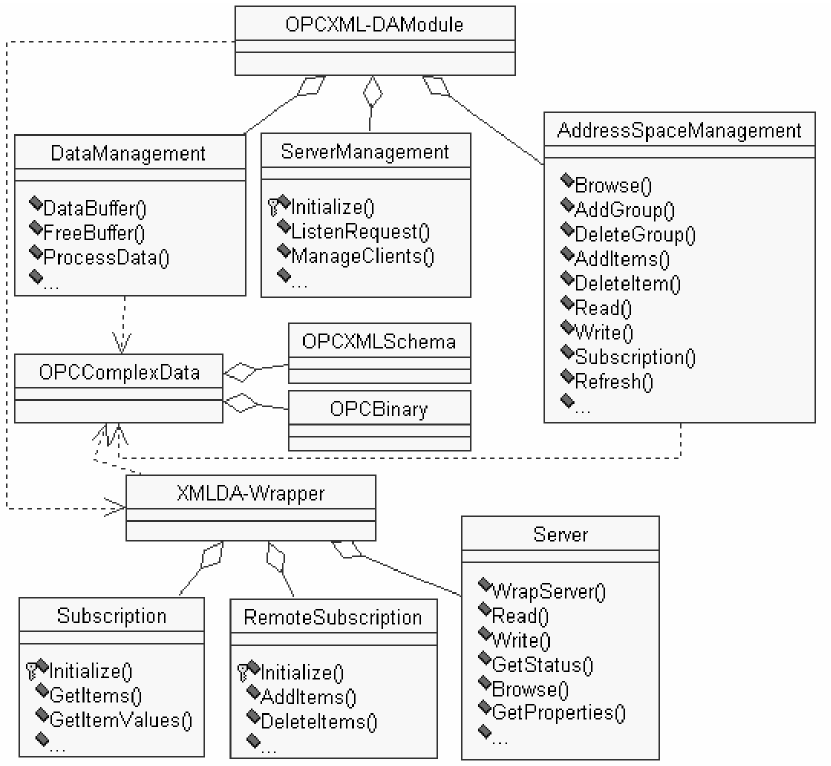

Fig. 5. The class diagram of OPCXML-DA module

as SOAP responses. This implementation also consists of the functions, which are used for performing the data transformation, data encryption/decryption, etc. By using the binary data representation with attachment technique, the SOAP messages are really reduced about size of the messages. Therefore, the bandwidth and system performances are truly improved much.

The Web Service listens to new connections, holds old connections, and handles their requests, etc. When a new connection request comes and the count of current connection is less than the max allowed number, this connection request will be accepted, otherwise rejected.

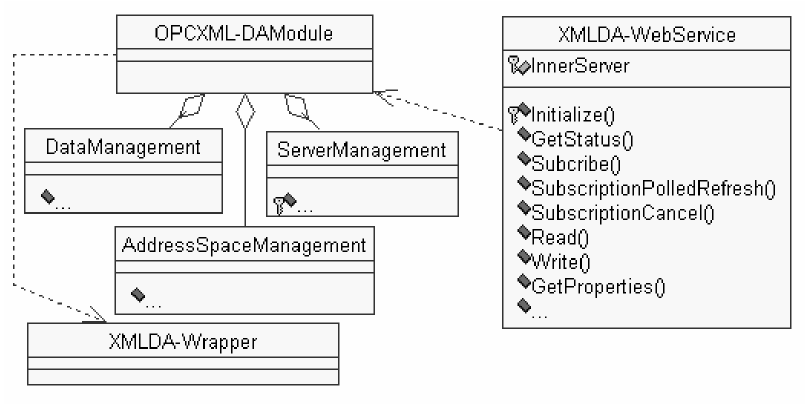

Fig. 6. The implementation of services for SOAP Request Handler

C. Illustrations

Our Web Service is created by basing on the ASP.Net Web Service template. To expose the implementation of our Web Services, we describe the Web Services' description in WSDL and the prototypes of these services corresponding to the C# language. As aforementioned in Section III-B, the services need to implement such as *Read*, *Write*, *GetStatus*, *Subscribe*, *SubscriptionPolledRefresh*, *SubscriptionCancel*, *GetProperties*, and *Browse*. In order to partly illustrate these services, the *Read* operation is represented in WSDL as the followings.

```
<s:element name="Read">
   <s:complexType>
     <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="Options"
                  type="s0:RequestOptions" />
       <s:element minOccurs="0" maxOccurs="1" name="ItemList"
                  type="s0:ReadRequestItemList" />
     </s:sequence>
   </s:complexType>
</s:element>
<s:complexType name="RequestOptions">
   <s:attribute name="RequestDeadline" type="s:dateTime" />
   <s:attribute default="true" name="ReturnErrorText"
                type="s:boolean" />
   <s:attribute default="false" name="ReturnDiagnosticInfo"
                type="s:boolean" />
   <s:attribute default="false" name="ReturnItemTime"
                type="s:boolean" />
   <s:attribute default="false" name="ReturnItemPath"
                type="s:boolean" />
   <s:attribute default="false" name="ReturnItemName"
                type="s:boolean" />
   <s:attribute name="ClientRequestHandle" type="s:string" />
   <s:attribute name="LocaleID" type="s:string" />
</s:complexType>
<s:complexType name="ReadRequestItemList">
```

```
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="Items"
                     type="s0:ReadRequestItem" />
        </s:sequence>
        <s:attribute name="ItemPath" type="s:string" />
        <s:attribute name="ReqType" type="s:QName" use="required" />
        <s:attribute name="MaxAge" type="s:int" />
     </s:complexType>
     <s:element name="ReadResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ReadResult"
                       type="s0:ReplyBase" />
            <s:element minOccurs="0" maxOccurs="1" name="RItemList"
                       type="s0:ReplyItemList" />
            <s:element minOccurs="0" maxOccurs="unbounded"
                       name="Errors" type="s0:OPCError" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

The *Read* operation that is generated from the WSDL into the corresponding C# language is shown as the following fragment.

```
[WebMethod(true)]
[SoapDocumentMethod(Namespace.XMLDA10_WEBSERVICE + "Read",
  RequestNamespace=Namespace.XMLDA10_WEBSERVICE,
  ResponseNamespace=Namespace.XMLDA10_WEBSERVICE,
  Use=System.Web.Services.Description.SoapBindingUse.Literal,
  ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)
]
public ReplyBase Read(
                RequestOptions      Options,
                ReadRequestItemList ItemList,
                out ReplyItemList   RItemList,
                [XmlElement("Errors")] out OPCError[]  Errors)
{
     // Code here
     // Get options from the request
     // Get a list of OPC Items that the client is interested in
     // ...
     RItemList = Request.GetResultList(replyList);
     Errors    = Request.GetErrors(errors);
     // Return the results
}
```

To continue illustrating the services using in our Web Service for control and monitoring systems, the *Subscribe* operation is used to subscribe a set of the items and then periodically polled using *SubscriptionPolledRefresh* operation. The *Subscribe* operation is described in WSDL as the followings.

```
     <s:element name="Subscribe">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="Options"
                       type="s0:RequestOptions" />
            <s:element minOccurs="0" maxOccurs="1" name="ItemList"
                       type="s0:SubscribeRequestItemList" />
          </s:sequence>
          <s:attribute name="ReturnValuesOnReply" type="s:boolean"
                       use="required" />
          <s:attribute default="0" name="SubscriptionPingRate"
```

```
                    type="s:int" />
        </s:complexType>
     </s:element>
      <s:complexType name="SubscribeRequestItemList">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="Items"
                     type="s0:SubscribeRequestItem" />
        </s:sequence>
        <s:attribute name="ItemPath" type="s:string" />
        <s:attribute name="ReqType" type="s:QName" use="required" />
        <s:attribute name="Deadband" type="s:float" />
        <s:attribute name="RequestedSamplingRate" type="s:int" />
        <s:attribute name="EnableBuffering" type="s:boolean" />
      </s:complexType>
```

The corresponding C# language for *Subscribe* operation from the described WSDL is represented as the following fragment.

```
[WebMethod(true)]
[SoapDocumentMethod(Namespace.XMLDA10_WEBSERVICE + "Subscribe",
 RequestNamespace=Namespace.XMLDA10_WEBSERVICE,
 ResponseNamespace=Namespace.XMLDA10_WEBSERVICE,
 Use=System.Web.Services.Description.SoapBindingUse.Literal,
 ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)
]
public ReplyBase Subscribe(
            RequestOptions    Options,
            SubscribeRequestItemList      ItemList,
            [XmlAttribute()]  bool  ReturnValuesOnReply,
            [XmlAttribute()][DefaultValue(0)]   int   SubscriptionPingRate,

            out SubscribeReplyItemList RItemList,
            [XmlElement("Errors")]  out OPCError[]    Errors,
            [XmlAttribute()]        out string        ServerSubHandle)
{
      // Code here
      // …
      // Return the results
}
```

The *SubscriptionPolledRefresh* operation is described in WSDL as follows:

```
<s:element name="SubscriptionPolledRefresh">
   <s:complexType>
     <s:sequence>
       <s:element minOccurs="0" maxOccurs="1" name="Options"
                  type="s0:RequestOptions" />
       <s:element minOccurs="0" maxOccurs="unbounded"
                  name="ServerSubHandles" type="s:string" />
     </s:sequence>
     <s:attribute name="HoldTime" type="s:dateTime" />
     <s:attribute default="0" name="WaitTime" type="s:int" />
     <s:attribute default="false" name="ReturnAllItems"
                  type="s:boolean" />
   </s:complexType>
</s:element>
```

So far, we have only represented several services to illustrate our implementation of Web Services for control and monitoring systems. Our illustrations have significantly provided more information for programmers or developers to develop the Web Services for the area of control and monitoring systems at the enterprise application levels in use today.

D. XML based Description Model

Framework-based application for industrial systems relies on an XML-based set of interface descriptions. These descriptions represent interaction schemas between specific applications and specific contexts. Conversations between the schemas are according to transformation rules. The application framework used content model to request data source and data formats for the values that have to be read out of the factory communication systems and their components. Since the XML Schemas contain information on the *data types*, *value ranges*, *display information*, *access paths*, and *parameters*. The application can determine and represent values and related information on the client easily. Therefore, using the WSDL files we can construct a standard XML-based application. An example for XML-based description model was represented in [15]; the XML-based content model consists of a distributed set of *XML files*, *schemas*, and *transformation rules*. In order to prevent the multi-definition, the files are linked together. Corresponding to the flexibility of XML-based descriptions, our services based on OPC XML-DA are generated according to WSDL descriptions. In addition, our Web Service can be configured by remote configuration clients with use of XML file and the data types are also described in WSDL file. Thus, these features guarantee the flexibility and effect of our system. Based on the XML-based descriptions, they indicate that our system is flexible when adding new services or changing the parameters or access paths or data types, etc.

IV. Data Connection and Representation

To discuss the publishing of real-time data in our system, we represent some considerations in Section IV-A. The evaluations of binary data representation are exposed in Section IV-B. Finally, comparisons of *XML Libraries* are then considerately discussed in Section IV-C.

A. Real-Time Data Publishing

In automation and control systems based on Web technologies in use today, the key problem is how to implement visual show of measured objects or refreshing of real-time data. To deeply demonstrate the ability to support high performance systems, we consider a feasible and high efficient solution that is real-time monitoring combined XML technologies. In our system, a Web browser to access the OPC XML-DA Server is used by thin client. This thin client obtains all monitoring views, curves, and reports in industrial systems. In order to separate views from real-time data, the Web browsers should save the monitoring view in XML file with attachment technique. Such static objects as *text*, *label*, *picture*, etc. are transmitted only once, and dynamic objects such as *pressure*, *liquid level*, etc. are shown for OPC Clients. Moreover, the OPC Clients will refresh when receiving respondent SOAP message. Therefore, the authorized OPC DA-XML Clients can monitor all field devices through the Web Server.

B. Binary Data Representation

Indeed, the only fast possibility to transmit the large amounts of data is to use the binary data representation rather than the XML data representation as aforementioned in Section II-C. In addition, different platforms and even different compilers of the same platform use different binary representations such as different bytes order for representing multi-byte data, different floating point formats, etc. Thus, universal standards should be chosen to transport data between different platforms. Therefore, we must investigate and compare the current available binary encoding standards. The fundamental approach to solve the bandwidth problem is using the binary data representation, which is integrated into XML such as BXML [16], BXSA [17], etc. Several proposals are available to satisfy these conditions such as SOAP message [18] with attachment or the HTTP message using XLink (XML Linking Language) [19]. The Web Service will respond to the XML-DA Clients a SOAP message with all the data replaced by

```
<ReadResponse>
  <ItemList>
    <Items ItemName="/Signal/Item" ValueType:"ArrayOfAnyType">
       <anyType xsi:type="xsdfloat">0.4554678</anyType>
       <anyType xsi:type="xsdfloat">0.5546789</anyType>
       <anyType xsi:type="xsdfloat">0.5884678</anyType>
       <anyType xsi:type="xsdfloat">0.8554678</anyType>
       <anyType xsi:type="xsdfloat">0.9559678</anyType>
       <anyType xsi:type="xsdfloat">0.9994678</anyType>
       <anyType xsi:type="xsdfloat">0.7554678</anyType>
       <anyType xsi:type="xsdfloat">1.7554678</anyType>
    </Items>
  <ItemList>                                                          (a)
--------------------------------------------------------------------------------------------
<ReadResponse>
   <ItemList>
     <Items ItemPath="S/Float" ValueType="BinaryStream">
        <BinaryStream type="typename" xlink=cid:SP1>
     </Items>
   </ItemList>
</>ReadResponse>
------
Content-ID: SP1                                                       (b)
...32 bytes Binary Data
```

Fig. 7. (a) A set of eight measurement values presenting in XML. (b) Replacement of these eight values by XLink reference with the binary data attachment

XLink references. To incorporate binary data into SOAP message, the WS-Attachment [20], [21] technologies are used. In addition, XLink is used to link different parts of compound message together. Addressing of multi-group can be done in XLink. Fig. 7 illustrates the use of the binary data presentation and WS-Attachment technologies.

C. *Comparisons of XML Libraries*

When design and implementation of an application system, we have to solve existing problems of weak ground. If there is new software available made by different manufacturers, the best thing to do is benchmarks. To achieve a better basic for a decision to select the best XML library, several benchmarks were performed to evaluate present XML libraries. There are many available XML library benchmarking projects such as XMark [22], XML Benchmark [23], SAX Parser Benchmark [24], etc. However, they are not very well applicable for the OPC XML-DA. In addition, they tested only one or two aspects of XML processing, with some predefined sequences of XML data. To provide fast and reliable OPC XML-DA solution, a fast multiplatform XML toolkit is required to process different types of XML files at high speed. Also, we have to investigate some supports including XML-binary Optimized Packing [25], XML Encryption [26], XML Signature [27], [28], XML Transformation [29], and some other.

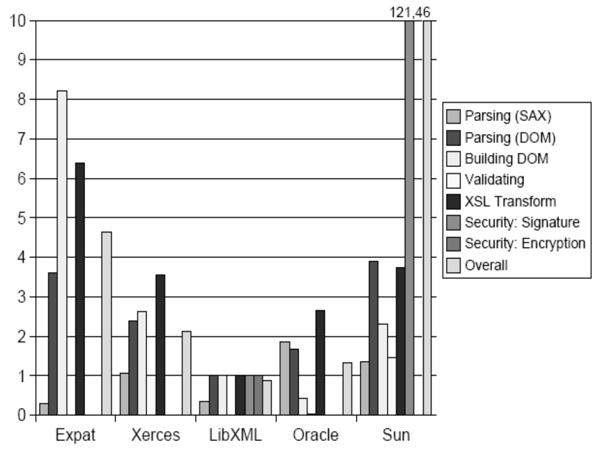

Fig. 8. Benchmark results of five different library combinations. The right most bar shows the overall performance of all processing stages. A blank bar indicates that the service is not available

The benchmark results of five different libraries are shown in Fig. 8 [30]. On some platforms, multiple compilers are available and should be compared, since the performances of the XML libraries are very important for the overall system performances. The performance tests can be divided into the following phases of XML processing such as Schema validation, XSL Transformation, XML Security, etc. As shown in Fig. 8, the *LibXML* is effectively high performance and it is selected to apply to our system.

V. Security Aspects

The security is very important position in application when using the Internet environment. In order to achieve required security criteria, the concepts and solutions developed for General IT system have to be needfully applied. Appropriate technologies like encryption, SSL (Secure Socket Layer) technology, HTTP-S (Secure HTTP), certificates and digital signature should be used. The solution for our system has to guarantee the effective security solutions to the OPC XML-DA. In addition reduce the bandwidth problems, the described solutions of improving performance raise new problems that are security problems. In the standard cases, the HTTP-S can be used to protect data. However, for multicasting data connections HTTP-S is not available and some other mechanisms must be used. A proposal is to use an authentication server that will use SSL private/public keys for authorization and generate symmetric session keys. Moreover, it also proposed to use the internal XML security approach for the control connections instead of the HTTP-S protocol, being described in XML Encryption [26], XML Signature specifications [27], [28], and XML Decryption [31]. This solution also provides several advantages over the HTTP-S approach were mentioned in [6]. The XML Distributed digital signature approach is represented in [32]. This solution is really optimal to reduce the data transfer between servers and clients, and ensures that clients are truly thin.

To guarantee the optimality of a security solution and the performance of our Web Service, we suggest applying the optimal security solution mentioned in [2], [32]. This solution reduces the data to transfer between the OPC XML-DA Server and Clients. Moreover, it is no necessary to build special environment for XML signature creation on the client side. The clients are truly thin in a Web application system that handles electronic application, electronic contacts or others. The XML distributed signature processing flow between the OPC XML-DA Server and Clients is represented as shown in Fig. 9.

Furthermore, the WSDL descriptions can be used to check and verify data passed to the services request. This reduces errors caused by inconclusive parameters. However, this feature does not prevent a user from sending faulty parameters. Other way for security is shown in [5] that solution used both symmetric secret and asymmetric key for security.

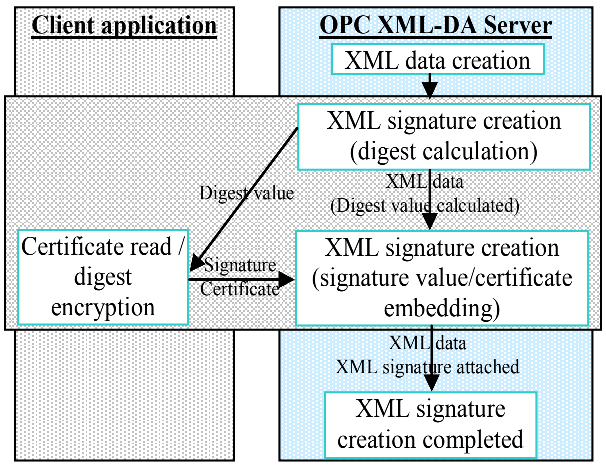
Client application
OPC XML-DA Server
XML data creation
XML signature creation (digest calculation)
Digest value
XML data (Digest value calculated)
Certificate read / digest encryption
Signature Certificate
XML signature creation (signature value/certificate embedding)
XML data XML signature attached
XML signature creation completed

Fig. 9. The XML distributed signature processing flow between the OPC XML-DA Server and Clients

VI. Performance Evaluations

Corresponding to a setup of the configurations for experimenting upon our system, we perform a number of tests by verifying the number of items in the OPC Clients. The setup for the performance tests consists of the OPC XML-DA Clients and OPC XML-DA Server with Windows XP Service Pack 2 as the following configurations:

- Intel Pentium® IV CPU ~ 2.66Ghz
- DDRAM for PC 3200 ~ 640MB
- Hard disk at least 5Gb of free space
- The OPC XML-DA Clients and Server PCs are communicated via a 10Mbps Ethernet network through a Hub device

We set out to measure the time taken for synchronous *Read*, *Write* operations and *Browse* operation to fetch a number of variables (i.e., the OPC Items) under various conditions. The experimental results of the time taken for synchronous *Read*, *Write* operations, and *Browse* operation to fetch a number of OPC Items in Data Access Server are shown in Fig. 10. These experimental results indicate that our design and implementation have improved the performance of the systems. As shown in Fig. 11, the comparison between XML-DA performance and COM-DA performance also indicates that XML-DA is about seven times slower than the COM-DA performance. It is easy to understand when DCOM data are sent in binary form. Our Web Service uses XML in which data are sent in SOAP message with attachment of the binary data representation form.

In fact, with the binary data representation and Web Service, however, the possibility of obtaining the real-time data in industrial systems is difficult to guarantee. Because real-time applications must receive timely data updates in packages that are small enough that they can be digested quickly. The mechanism of XML Web Service works well for business applications such as CMMS, ERP, and EAM. In the real-time applications, this approach works poorly as might be expected. However, we

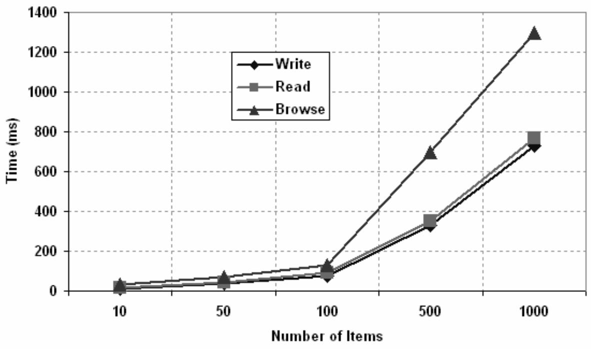

Fig. 10. The results of the time taken for synchronous *Read*, *Write* operations and *Browse* operation to fetch a number of OPC Items in Data Access Server

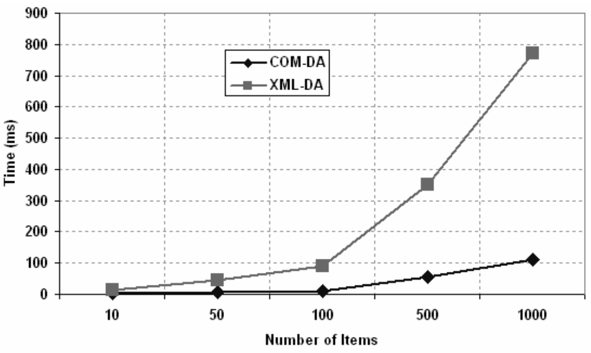

Fig. 11. A comparison between XML-DA performance and COM-DA performance indicates that XML-DA is about seven times slower than COM-DA. However, the data rate of 1000 items in 770ms is acceptable for many real systems in industry today (*Read* operation)

can improve the system performance by using several mechanisms. In order to improve real-time performance in the whole system, several complementary mechanisms should be implemented as the followings:

- Firstly, "*Subscription-Refresh*" model is adopted in OPC Clients. The OPC XML-DA Server need only to send the changed data. This model decreases net flow enormously and enhances communication performance on network.
- Secondly, with the development of network architectures the high speed Ethernets (e.g., 100Mbps, 1Gbps, etc.) gradually become common in LAN and WAN. In addition, the bandwidth resources are increasing fast because of the development of the switch technologies. Therefore, the system performance will be improved as well as possible.
- Furthermore, such attributions in the client's requests as *RequestSamplingRate*, *EnableBuffering*, and *DeadBand* could be set to needfully optimize performance of Web Server to meet the expectations of real-time data transmission, refreshing, view, etc.

However, our Web Service has a good performance and has significantly improved the limitation of the performances of the XML textual representation in the Web applications. Thus, our system performance is acceptable for many control and monitoring systems in nowadays.

VII. Conclusions and Future Work

In this paper, we have already introduced the design and implementation of Web Service by using the OPC XML-DA and OPC Complex data for industrial systems at the enterprise levels, especially for slow process monitoring and control systems. We have proposed a means to implement the OPC XML-DA Server supporting the OPC Complex data. Therefore, the OPC XML-DA Clients can read and decode any type of data from the hardware I/O devices on the plant floor or OPC Servers. The implementation can be characterized to be extremely flexible and reusable. To overcome the performance limitations of the pure XML textual representation, we have significantly proposed to use the binary data representation in Web Service. Besides, we have investigated and adequately compared the *XML Libraries* to choose XML technology as a candidate for our system. And, the *LibXML* should be used to resolve the system requirements (e.g., acceptable performance, any platform, compatible with middleware, etc.). However, the use of the binary data representation with its significant performance improvements in only limited applications or situations was mentioned. Our simulation results indicate that the performance of our system has an acceptable performance and is acceptable for many industrial systems in use today.

In addition, the security aspects are carefully discussed and analyzed in our works. These aspects provide seamless security solutions in big systems that are communicated by using the Internet environment. They also provide more security information for technical-level readers. Moreover, we also suggested using the optimal security solution based on the XML Distributed signature in our system. This solution can manage the access right on the OPC Items' level and reduces the data processing flows between the OPC XML-DA Server and Clients. Additionally, this solution also truly ensures the thin clients.

However, the performance of a system in Web applications is depended on the size of a message transmitted on the network. So a variability of our system performance related to the size of the data in the SOAP message will be investigated in the future work. In addition to apply our Web Service in the Internet environment, the system performance evaluations should be carefully investigated.

Acknowledgments

The authors would like to thank the Ministry of Commerce, Industry, and Energy and Ulsan Metropolitan City, which partly supported this research through the Network-based Automation Research Center (NARC) at the University of Ulsan. The authors also wish to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] D. W. Holley, "Understanding and Using OPC for Maintenance and Reliability Applications," *Journal of IEE Computing and Control Engineering*, pp. 28-31, February-March 2004.

[2] V. V. Tan, D. S. Yoo, and M. J. Yi, "Design and Implementation of Web Service by Using OPC XML-DA and OPC Complex Data for Automation and Control Systems," *Proceedings of the 6th IEEE International Conference on Computer and Information Technology*, CIT'06, p. 263, September 2006.

[3] OPC XML Data Access Specification Version 1.01, [Online]. Available: http://opcfoundation.org/, December 2004.

[4] OPC Complex Data Specification version 1.0, [Online]. Available: http://opcfoundation.org/, December 2003.

[5] S. Chilingargyan and W. Eppler, "High Speed Data Exchange Protocol for Modern Distributed Data Acquisition Systems based on OPC XML-DA," *Proceedings of the 14th IEEE-NPSS Real-time Conference*, pp. 352-356, June 2005.

[6] W. Eppler, A. Beglarian, S. Chilingarian, S. Kelly, V. Hartmann, and H. Gemmeke, "New Control System Aspects for Physical Experiments," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 482-488, June 2004.

[7] M. Wollchlaeger, P. Nweumann, and Th. Bangemann, "Web Service for Remote Maintenance of Fieldbus based Automation Systems," *Proceedings of the IEEE International Conference in Africa*, pp. 247-252, October 2002.

[8] OPC Data Access Specification, [Online]. Available: http://opcfoundation.org/, December 2004.

[9] G. Eisenhauer, F. Bustamante, and K. Schwan, "Native Data Representations: An Efficient Wire Format for High Performance Computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 13, no. 12, pp. 1234-1246, 2002.

[10] OPC Historical Data Access Specification, [Online]. Available: http://www.opcfoundation.org/, December 2003.

[11] OPC Alarms and Events Access Specification, [Online]. Available: http://www.opcfoundation.org/, October 2002.

[12] Z. Jia and X. Li, "OPC-based Architecture of Embedded Web Server," *Proceedings of the ICESS*, LNCS-3605, Springer-Verlag, pp. 362-367, 2005.

[13] Web Services Description Language (WSDL) Version 2.0, March 2006, [Online]. Available: http://www.w3.org/TR/wsdl20/

[14] XSL Transformation (XSLT) Version 1.1, 2001/8/24, [Online]. Available: http://www.w3.org/TR/2001/WD-xslt11-20010824/

[15] M. Wollchlaeger and T. Bangemann, "XML based Description Model as a Platform for Web-based Maintenance," *Proceedings of the IEEE Conference on Industrial Informatics*, pp. 125-130, June 2004.

[16] C.S. Bruce, "Cubewerx Position Paper for Binary XML Encoding," [Online]. Available: http://www.cubewerx.com/main/HTML/Binary_XML_Encoding.html

[17] K. Chiu, T. Devadithy, W. Lu, and A. Slominski, "A Binary XML for Scientific Application," *Proceedings of the 1st International Conference on e-Science and Grid Computing*, pp. 336-343, December 2005.

[18] SOAP (Simple Object Access Protocol) Version 1.2, W3C Recommendation 2003/6/24, [Online]. Available: http://www.w3.org/TR/SOAP

[19] XLink (XML Linking Language) Version 1.0, 2001/6/27, [Online]. Available: http://www.w3.org/TR/xlink/

[20] SOAP message with Attachments, December 2000, [Online]. Available: http://www.w3.org/TR/SOAP-attachments

[21] J. H. Gailey, Using Web Services Enhancements to Send SOAP Messages with Attachments, Feb. 2003, [Online]. Available: http://msdn2.microsoft.com/en-us/library/ms996944.aspx

[22] XMark, an XML benchmark project, [Online]. Available: http://monetdb.cwi.nl/xml/index.html
[23] XML Benchmark, [Online]. Available: http://www.sosnoski.com/opensrc/xmlbench/results.htm
[24] SAX Parsers Benchmark, [Online]. Available: http://piccolo.sourceforce.net/bench/html
[25] XML-binary Optimized Packing, W3C Recommendation, 2005/01/25, [Online]. Available: http://www.w3.org/TR/2005/REC-xop10-20050125/
[26] XML Encryption Syntax and Processing, W3C Recommendation 2002/10/10, [Online]. Available: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[27] XML-Signature XPath Filter 2.0, W3C Recommendation, 2002/11/08. [Online]. Available: http://www.w3.org/TR/xmlenc-core
[28] XML Signature and Processing, W3C Recommendation, 2002/2/12. [Online]. Available: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
[29] XSL Transformation (XSLT) Version 1.1, W3C Recommendation, 2001/8/24. [Online]. Available: http://www.w3.org/TR/2001/WD-xslt11-20010824
[30] S. Chilingarian, Fast, Multiplatform XML Toolkits Comparison. Functionality, Speed and Memory Usage, [Online]. Available: http://prdownloads.sourceforge.net/xmlbench/features.pdf, http://prdownloads.sourceforge.net/xmlbench/benchmark.pdf
[31] Decryption Transformation for XML Signature, W3C Recommendation, 2002/10/10. [Online]. Available: http:www.w3.org/TR/xmldenc-decrypt/
[32] Miyauchi, "XML Signature/Encryption – The Basic of Web Service Security," *NEC Journal of Advance Technology*, vol. 2, no. 1, pp. 35-39, Winter 2005.
[33] U. Katsuji, S. Shinichi, and W. Hidehiko, "A Prototype Embedded XML-DA Server and its Evaluation," *Proceedings of the SICE-ICASE International Joint Conference*, pp. 4331-4336, October 2006.

Vu Van Tan was born in Haiduong province, Vietnam, in 1981. He received the B.Eng. degree in Information Technology from the Hanoi University of Technology, Vietnam, in 2004. He has worked as design and analysis engineer in KhaiTri Software Company, Vietnam, for one year. He is currently a Ph.D student in the School of Computer Engineering and Information Technology, University of Ulsan, Republic of Korea. He joined in the Applied Software Engineering Lab, University of Ulsan, in 2005. His main interests are software engineering, software for automation systems, Internet technologies for industrial automation systems, and real-time communication systems.

Dae-Seung Yoo received the B.S. and M.S. degrees in Computer Engineering and Information Technology from the University of Ulsan, Republic of Korea in 1998 and 2001, respectively. Now he is a guest professor in the School of Computer Engineering and Information Technology, University of Ulsan. He is a member of KIISE, KIPS, and IEICE. His main interests are software engineering, software for automation systems, and Internet technologies for industrial automation systems.

Myeong-Jae Yi received the B.S. degree in Computer Science from the Seoul National University, Republic of Korea in 1987. He also received the M.S. and Ph.D degrees in Computer Science from the Seoul National University in 1989 and 1995, respectively. He is currently a professor in the School of Computer Engineering and Information Technology, University of Ulsan, Republic of Korea. He is also a vice Director of NARC (Network based Automation Research Center) at the University of Ulsan and is a member of KIISE, ACM, and IEEE. His main interests are software engineering, software for automation systems, and Internet technologies for industrial automation systems.