Development of an OPC Client-Server Framework for Monitoring and Control Systems

Vu Van Tan* and Myeong-Jae Yi**

Abstract—In this article, the current technological state of OPC (Openness, Productivity, and Collaboration; formerly "OLE for Process Control") standards and the problem statement of these OPC standards are discussed. The development of an OPC client-server framework for monitoring and control systems is introduced by using the new OPC Unified Architecture (UA) specifications, Service Oriented Architecture (SOA), web services, XML, etc. The developed framework in turn minimizes the efforts of developers in learning new techniques and allows system architects and designers to perform dependency analysis on the development of monitoring and control applications. The potential areas of the proposed framework and the redundancy strategies to increase the efficiency and reliability of the system are also represented according to the initial results from the system that was developed by the Visual Studio 2008 and OPC UA SDK.

Keywords—OPC, OPC UA SDK, Monitoring and Control, Redundancy, Unified Architecture

1. INTRODUCTION

In general, a monitoring and control system can be characterized as a distributed and integrated monitoring, control, and coordination system with partially cyclic and event-based operations. Its control functions can be divided into continuous, sequential and batch control. In addition to control functions, such a system has other functions including performance monitoring, condition monitoring, abnormal situation handling, and reporting [38, 16]. Information achieved from monitoring and control systems can be used for the condition based maintenance system that uses this information to make decisions such as the loop and sending out an alarm or event, to re-identify a system, and to monitor a system [24, 11].

A few competing protocols such as FOUNDATION fieldbus, PROFIBUS, DeviceNet, ControlNet, and Ethernet/IP can be found in automation systems [34]. Different communication solutions adopted today share a historical problem related to the fact that data from different systems may have different formats and different communication protocols. This is a key factor when, for instance, drives are connected to a Supervisory Control And Data Acquisition (SCADA) system. Software vendors creating process monitoring, control, and data management

※ This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MEST) (NRF-2010-0016252). The authors also would like to thank the anonymous referees for their valuable comments and suggestions.
Manuscript received October 15, 2010; first revision January 25, 2011; accepted March 7, 2011.
Corresponding Author: Myeong-Jae Yi
* Network Based Automation Research Center (NARC), University of Ulsan, San-29, Moogu-2 dong, Namgu, Ulsan 680-749, South Korea (tanvv2905@gmail.com)
** School of Computer Engineering and Information Technology, University of Ulsan, San-29, Moogu-2 dong, Namgu, Ulsan 680-749, South Korea (ymj@mail.ulsan.ac.kr)

systems must develop individual I/O drivers for each protocol. Developing unique drivers for each type of the plant floor (or shop floor) is not only time consuming and inefficient, but also inherently adds additional risks to the successful and timely completion of a project. For these reasons, OPC technology has been developed as a middleware technology for integration and interoperation since 1996 by Fisher-Rosemount, Intellution, Intuitive Technology, Poto22, Rockwell, and Siemens AG [28].

The classic OPC technology uses a client-server approach for the information exchange [13]. An OPC server encapsulates the source of process information like a device and makes the information available via its interfaces. An OPC client connects to the OPC server and can access and consume the offered data. Although the OPC technology comes from industries, which still makes a lot of researcher suspicious [24], but the classic OPC technology like OPC Data Access (DA) [33] or OPC Alarms and Events (AE) [30] is not a new and fancy name for something already used for many years. It can be a possibility for many universities, research departments, and institutes to get products into the market (e.g., Advosol Inc. [1], MatrikonOPC [20], etc.).

A number of approaches have been recently done on developing methods or guidelines for the design and development of monitoring and control systems using the classic OPC technology. However, current approaches are still far systematic design methods or an ultimate solution due to a late indication of changes in the production environments, a delay of the implementation of changed production plans, and changes in technology [8, 38]. In addition, they have only resulted in small-scale capability or specific-applications in the domain of monitoring and control applications as studied by Eppler et al. [10], Chilingargyan and Eppler [5], Jia and Li [15], Usami et al. [35], Khailgui et al. [17], Torrisi and Oliveira [34], Venzke et al. [36], Sahin and Bolat [23], Han et al. [12], Yang et al. [38], Wang et al. [37], etc. Three types of the data such as current data, historical data, and alarms and events are not related to others in the address space of the server. The OPC Foundation now is working on supporting XML and web services, as well as in the OPC Unified Architecture (UA) specifications [29], so that Internet-based monitoring and control with the leverage of XML and web services will become a reality [8, 18].

The Service Oriented Architecture (SOA) is a new architectural paradigm used for distributed computing and services sharing [25]. It has evolved from object oriented and component based architecture to being able to develop networks of collaborating applications. It also utilizes services with autonomous platform-independent computational elements that can be described, published, discovered, and accessed over the Internet by standard protocols [6, 3, 22].

This research attempts to provide state-of-the-art OPC standards and the current situation and problem statement of such standards. The OPC client-server framework, which is based on the SOA and new OPC UA specifications, was proposed and developed. This framework is developed and implemented as an integration solution for the monitoring and control of devices and the intention of condition based maintenance. The initial results from the performed simulation to demonstrate the good performance of the proposed framework are provided. The potential areas of the proposed architecture and the redundancy strategies to increase the efficiency and reliability of the system are discussed.

This paper is organized as follows: the next section provides state-of-the-art OPC technology. In Section 3, the current situation and problem statements of OPC standards are explained in detail. The concept of an OPC UA client-server framework is introduced in Section 4. Section 5 discusses the potential areas of the proposed architecture based on the performance of the proposed framework and the redundancy strategies, respectively. Finally, Section 6 concludes some remarks and future works.

2. WHAT IS OPC?

Software applications like Distributed Control System (DCS) and Enterprise Resource Planning (ERP) can be used with SCADA systems for a range of different purposes. SCADA software systems are used in the automation industry for the applications of industrial measurement, monitoring, and control. Components for SCADA software systems were designed and developed by a single software or hardware company for a specific system or specific domain, which has a lack of generally accepted interfaces. Thus, these components cannot be applied to systems manufactured by other companies. It is essential to consider how to design a set of standard components for the automation industry.

An example of conventional communication architecture between three different software applications and three different process control devices in the automation industry is shown in Fig. 1. Each software application in the system has to have a driver for each device in order to be able to communicate with each other. It is easy to recognize several disadvantages of such a conventional architecture like complexity, high cost, inefficiency, etc. Therefore, the hardware and software companies should develop new solutions for this. In addition, different communication solutions available today share a history problem that data from different systems have different formats and different communication protocols. Software vendors creating process monitoring, control, and data management systems have to develop individual I/O drivers for each protocol. Developing unique drivers for each type of the plant floor is not only time consuming and inefficient, but also inherently adds additional risks to the successful and timely completion of a project. For those reasons, the OPC Foundation was established.

OPC technology is the technological basis for the convenient and efficient linking of automa-

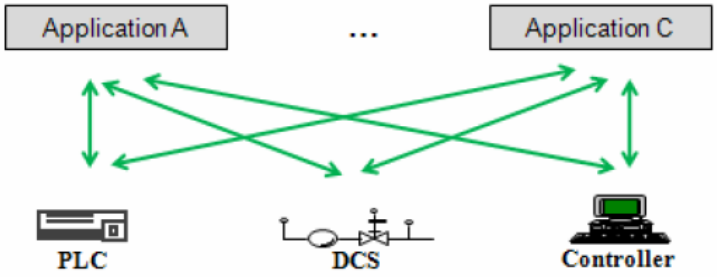

Fig. 1. Conventional communication architecture

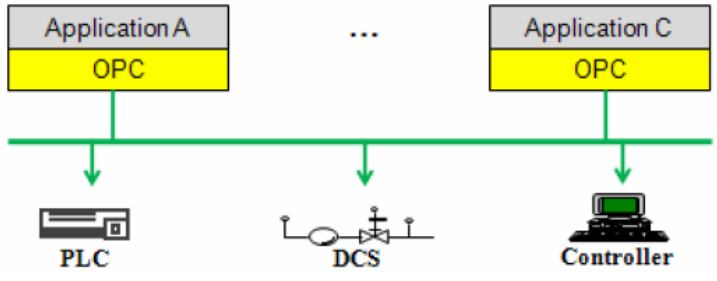

Fig. 2. Communication architecture based on the OPC standard

tion components with control hardware and field devices. It is also for communication among various data sources, either devices on the plant floor, or a database in a control room. A communication architecture using the OPC standard is shown in Fig. 2, which provides a means to gather data from a data source. These data are communicated with any client application using a standard and efficient method. The most diverse OPC components of different manufacturers can work together with no additional programming for the adaption of the interface between components [21, 13].

2.1 Current OPC Specifications

OPC specifications include important industrial issues like Data Access (DA), Alarms and Events (AE), Historical Data Access (HDA), Batch, Security, XML, Complex Data, and Data eXchange as shown in Fig. 3 [21]. These specifications now are accepted as industrial standards for the automation industry. The OPC UA specifications defining a vendor and protocol independent server-client model have been released and will become the IEC 62541 standard [21, 29]. In general, every specification is used for a specific field of industrial applications and is based on a model that corresponds to the field of such applications.

Since the introduction of the new OPC generation, called Unified Architecture (UA), the OPC specifications have been divided into "classic" OPC and OPC UA, respectively [29, 19]. The classic OPC technology is based on Component Object Model (COM)/Distributed COM (DCOM) technology from Microsoft. The advantage of using COM/DCOM technology was the reduction of the specification work to the definition of different APIs for different specialized needs without the requirement to define a network protocol or mechanism for inter-process communication [21]. However, the main disadvantages are Windows-platform dependency and DCOM issues when using remote communication. The OPC UA standard uses XML and web services as base technologies for transferring data from field devices to enterprise applications via the Internet. It defines only one address space for process data, alarms, historical data, and programs.

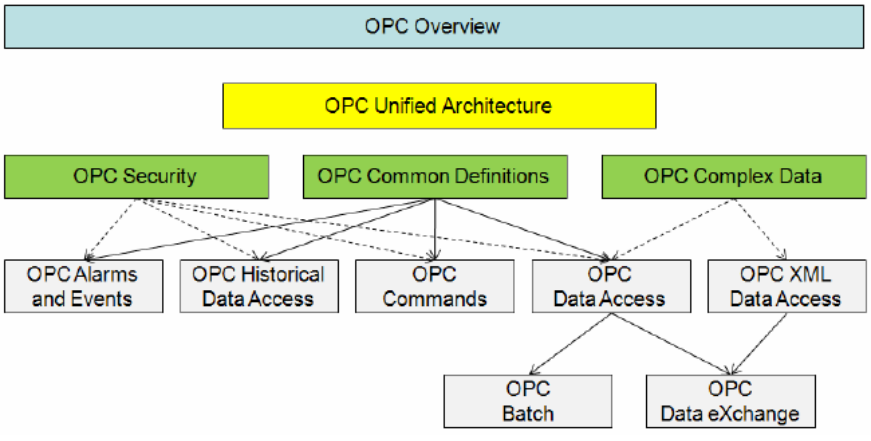

Fig. 3. Available OPC specifications

3. THE CURRENT STATE OF OPC TECHNOLOGY

3.1 COM/DCOM Technology

COM technology developed by Microsoft enables software components to communicate. COM is used by developers to create re-usable software components, to link components together to build applications, and to take advantages of windows services. COM objects can be created with a variety of programming languages [4]. This defines interaction mechanisms between objects including data objects and application objects. Data on these objects are stored and can be accessed locally. Interfaces that are a collection of functions are used for communication between objects. The family of COM technologies includes COM+, DCOM, and ActiveX Control [4]. Classic OPC technology is based on COM/DCOM technology, but the COM/DCOM technology can detect timeouts which can lead to unreliable data transmission such as hardware problems, overload networks, networks based on satellite links, etc [24].

Classic OPC technology still has some limitations due to the use of COM/DCOM technology. These limitations are as follows: (i) it is only for Windows platforms; (ii) DCOM can be used for the applications over the Internet, but firewall authentication problems are not easy to resolve; and (iii) data exchange between devices on the plant floor and enterprise applications is an issue that needs to be tackled.

3.2 Web Services

Web services technology is software components based on Web Services Description Language (WSDL) [7]. These components are capable of being accessed via standard network protocols such as SOAP over HTTP/HTTPS. Application developers can quickly create and use web services by employing existing tools and frameworks, and web servers also supply the essential infrastructure to exploit web services. XML and web services today are accepted in the industry and business [39, 14]. The OPC Foundation is working on supporting XML and web services for developing industrial applications as introduced in the OPC UA specifications [29].

Web services have a few disadvantages: Only the method of Poll-Report-By-Exception is provided and the driver reports the changes for the time intervals where the services were disconnected [25]. Therefore, this method works well for applications where data is collected from remote sources, but does not require real-time data updates. However, the size of XML messages is much larger than the size of DCOM messages when carrying out the same information. The solution of using binary data encoding can be used for web services to improve the performance of the system [29, 10, 5].

3.3 OPC Unified Architecture

The OPC UA standard is the next generation technology for secure, reliable, and interoperable transport of raw data and preprocessed information from the plant floor or shop floors to production planning or ERP systems [29]. It extends existing OPC standards with fundamental features such as platform independence, scalability, high-availability, Internet capability, and many more. It defines services that should be provided by servers for clients and also defines how servers can indicate which services they provide. One of the key problems with the new standard is that implementing it can be quite challenging. To do this, the OPC Foundation has taken many steps to guarantee that the implementation of such a standard would be relatively

straightforward and easy process.

To facilitate the adoption of this new standard and to reduce the barrier to entry, an OPC-UA Software Development Kit (SDK) is being developed. The SDK is the entry point to jump-start the existing applications and makes them OPC-UA-enabled. The SDK consists of a series of application programming interfaces and sample code implementations. To that end, the OPC UA specifications are written to be platform-agnostic and, for that reason, the SDK comes in different flavors to facilitate the adoptions on different platforms. The .NET, ANSI C, and Java implementation samples will be provided [28].

This unified standard intends to enable enterprise interoperability and expects to solve enterprise integration challenges. It is clear that this standard does not provide everything needed for interoperability from the enterprise-IT perspective, but the impact is expected to be considerable.

3.4 Security Issue

Security is becoming an important issue in the automation industry due to the collaboration of business networks and process networks. Devices are now being connected via such communication systems and Programmable Logic Controllers (PLCs) can be easily programmed via a communication network. Data and events from industrial systems are sent to operator systems and enterprise systems via the Ethernet and Internet. The problems with malicious network attacks will be carried into the process area, and viruses and worms will corrupt the system from PLC to field devices. For the classic OPC standards (e.g., OPC Data Access), an OPC server might implement one of three levels of security [9]: (i) Disabled security - no security; (ii) DCOM security - launch and access permissions to OPC servers are limited to selected clients; or (iii) OPC security - the OPC server serves as a reference monitor to control access to vendor-specific security objects exposed by the OPC server.

The OPC security specification covers only server/object access control, but is not concerned with confidentiality and integrity during transmission. However, additional security settings as well as defined in the OPC security specification have been implemented in only a few products on the market [21]. The OPC UA standard has a scalable security concept based on W3C standards and includes user authentication, digital signatures, and encryption for the exchange messages [29, 19]. Implementation, evaluation, and performance depending on this standard are now a challenge for researchers and developers.

3.5 Discussion

Industrial systems now need to be independent of any operating system and platform. The new OPC UA technology will surely rise as the eventual winner and the OPC DA technology still continues to dominate applications used in operations that this standard focuses on solving real-time data transfer requirements. Therefore, all of the OPC standards (e.g., OPC UA [29], OPC XML-DA [32], OPC HDA [31], and OPC DA [33]) complement each other and they will continue to do more in the future. The OPC UA technology can be used on all levels of industry processes, from sensors through SCADA systems to MES/ERP systems. More than 600 products have been based on the OPC specifications because these OPC specifications are now accepted as industrial standards [21]. However, designing, developing, and deploying industrial automation applications based on the new OPC UA specifications are very challenging to system architects, designers, and developers due to the complexity and difficulty of related control

and monitoring decision tasks and information systems.

4. Proposed OPC Client-Server Framework

4.1 Overview of the Framework Architecture

With the introduction of web services at the level of production devices or facility automation, there are still big problems in communication between device-level SOA and the one that is used in back-end systems [8]. These problems can be overcome by using a middleware between the back-end applications and the services, which are offered by devices, service mediators, and gateways. At least two different ways can be used to couple networked embedded devices with enterprise services: (i) direct integration of device-level services in business processes and (ii) exposure of device functionalities to the application layer via a middleware layer [16].

The framework architecture to expose device-level services from plant floors to enterprise systems using OPC technology is proposed as shown in Fig. 4. The service provided by a device at the device level offers a given functionality to the service user, and provides a well-defined interface in which the service can be invoked by the user from the OPC UA server. It runs transparently from the viewpoint of the user. To support OPC UA clients that can access existing OPC servers such as OPC DA servers [33], OPC HDA servers [31], and OPC AE servers [30], the wrappers to map the address space of these servers into the address space of OPC UA server have been developed. However, many of innovations and advantages of the OPC UA technology are lost such as the uniform access to process data, historical data, and alarms in one server address space, programs, type information, and structured data types. The wrapper also represents an additional conversion layer that reduces the transmission rate considerably [19].

4.2 Information Model

To provide better integration of alarms and events in the address space of an OPC UA server, three different types of data such as current data, historical data, and alarms and events are en-

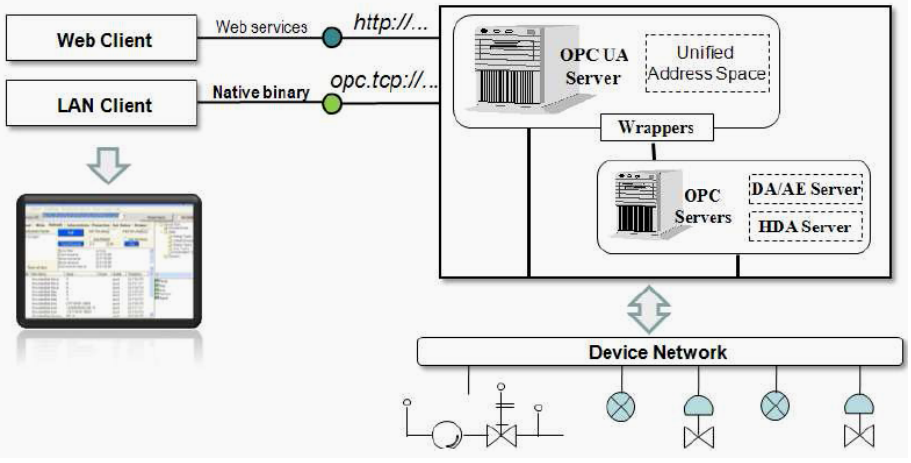

Fig. 4. The proposed architecture of an OPC client-server framework for monitoring and control

abled to be accessed by a single OPC UA server based on a unified object model [27]. In addition, commands or methods from device objects can be represented as method services in the address space of the server. For example, the methods like "Start" and "Stop" in a motor can be invoked by the client to start or stop the motor. Each method specifies the input arguments and the output arguments. A unified object model is developed for three types of data: (i) current and historical data can be stored in variables; (ii) the commands to control devices at the process plants can be considered as method services for execution; and (iii) the occurrence of an alarm or event from hardware devices can be considered as an event service [26].

4.3 Server Implementation

The steps to be implemented for the development of an OPC UA server are as follows: (i) the necessary libraries are loaded and static variables are instantiated; (ii) the configuration of application for the necessary certificates and the identification of the application; (iii) the construction of address space in the OPC UA server; and (iv) the configuration and opening of endpoints that a server can create several endpoints with different security policies and modes for communication with the clients [21, 29].

The OPC SDK provided by the OPC Foundation is implementing the features that are common to all OPC UA servers. It is designed to be extensible and allows the integration of data and events from different sources. The following functionalities have to be implemented for a production server: (i) exposing one or more endpoints; (ii) security environments like authentication, authorization, certificates maintenance, security policies and modes; (iii) configuration maintenance and connection handling (e.g., the management of endpoints); and (iv) address space management; (v) underlying real-time process interoperability (e.g., reading and writing data to devices), etc.

The StandardServer class is the root class for the development of a server application. It provides implementations for most of the UA services. Therefore, developers can customize these implementations by creating a subtype of the StandardServer and can override the appropriate methods. After initializing the endpoints, the "Start" method creates all of the manager objects, which handle different aspects of the server functionality and stores them in an instance of the ServerInternalData object. For example, creating a session in the OPC UA server application is

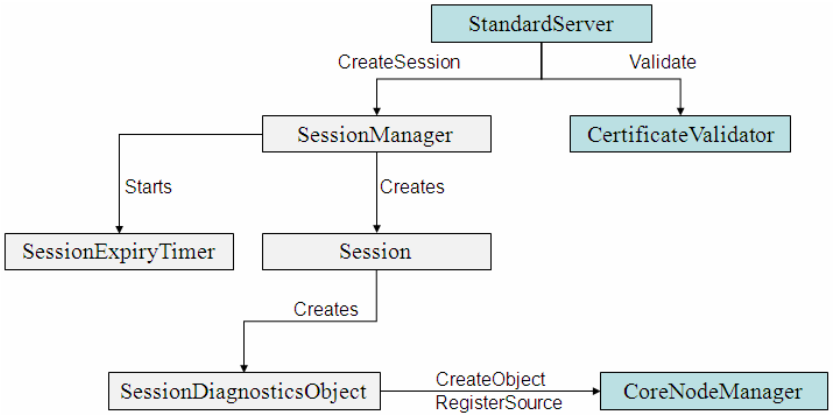

Fig. 5. Creating a session from the viewpoint of the StandardServer

shown in Fig. 5: (i) when an OPC UA client creates a session, the OPC UA server must validate the client application instance certificate and the client software certificates; (ii) the session manager then creates, monitors, or closes sessions and also validates user identity; and (iii) the session impersonates the user, validates requests, and manages continuation points.

Based on the StandardServer class, the SampleServer class, for example, can be inherited and implemented in the C# language as follows:

```
public class SampleServer : StandardServer
{
protected override void LoadConfigration()
{
// Load the configuration file for the server.
}
protected override MasterNodeManager
CreateNodeManager(ServerConfiguration configuration)
{
// Create the master node manager for the server.
// Get the list of available wrappers
// e.g., COM DA wrappers, AE wrappers, etc.
}
protected override ServerProperties GetServerProperties()
{
// Get the properties of a server.
}
protected override void OnNodeManagerStated()
{
// Create the object and add it to the address space
// For example
m_boiler1 = new Boiler(this);
m_boiler1.CreatInstance(
Objects.ObjectFolder,
Opc.Ua.ReferenceTypes.Organizes,
false,
new QualifiedName("Boiler 1", 1));
m_boiler1.Register();
...
}
...
public override ResponseHeader Read(
RequestHeader requestHeader,
int maxAge,
TimestampsToReturn timestampsToReturn,
ReadValueIdCollection nodesToRead,
out DataValueCollection values,
out DiagnosticInfoCollection diagnosticInfos)
```

```
{
// Read data from the server.
}
public override ResponseHeader Write(
RequestHeader requestHeader,
WriteValueCollection nodesToWrite,
out StatusCodeCollection results,
out DiagnosticInforCollection diagnosticInfos)
{
// Write data to the server.
}
...
}
```

The Read and Write services not only allow us to read and write the values of variables, but are also used in a generic way to read and write the attributes of nodes to access metadata in the address space of an OPC UA server. The Read service is used to read one or more attributes of one or more nodes [29]. It also allows for reading subsets or the single element of array values and to define a valid age of values to be returned to reduce the need for devices. The request parameters for the Read service are *MaxAge*, *TimestampsToReturn*, and *NodesToRead*. The response parameters for the Read service are *Value*, *StatusCode*, *SourceTimestamp*, and *ServerTimestamp* [19]. The implementation of the Read service in the C# language can be shown as follows:

```
...
diagnosticInfos = null;
DataValue value = new DataValue();
value.Value = Int32.MaxValue;
value.SourceTimestamp = DateTime.UtcNow;
values = new DataValueCollection(nodesToRead.Count);
foreach (ReadValueId valueId in nodesToRead)
{
values.Add(value);
}
return new ResponseHeader();
```

The Write service is used for writing one or more attributes of node(s) and subsets or single element of array values. It is optimized for bulk write operations but not for writing single value. The request parameters for the Write service are *NodeId*, *AttributeId*, *IndexRange*, *Value*, *StatusCode*, *SourceTimestamp*, and *ServerTimestamp*. The response parameters for such a service are a list of result status codes for each Write operation (e.g., Results[]).

From the viewpoint of the StandardServer, monitored items can be created and added for a subscription as shown in Fig. 6(a) and reporting data changes from the underlying systems to the monitored items is shown in Fig. 6(b). A subscription maintains a list of monitored items which must be polled for notifications to report. The NodeManager handles subscriptions for data values and the EventManager handles subscriptions for events. The NodeManager has to create a

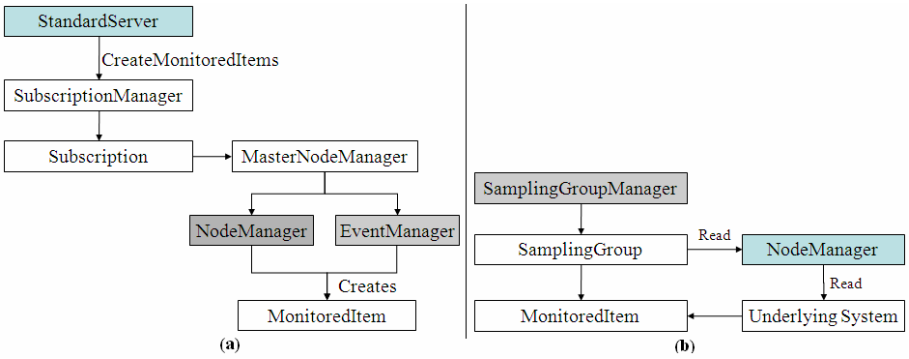

Fig. 6. (a) Creating monitored items in the viewpoint of StandardServer. (b) Reporting data changes from the underlying systems to monitored items.

SamplingGroup for polled values where the SamplingGroup periodically reads the values and updates them to the MonitoredItem.

4.4 Client Implementation

For facilitating the development of OPC UA client applications for specific industrial applications, the OPC UA client framework is proposed and developed by handling the standard tasks in which all OPC UA clients need to do. These tasks for a development like this include: (i) sending the session keep-alive requests, (ii) keeping the track of current status for subscriptions and monitored items, (iii) managing a client-node-cache and data publish issues, (iv) processing and caching incoming data change and event notifications, and (v) saving and restoring the session state. The necessary classes for the development of OPC UA client applications can be seen in Fig. 7.

All of the OPC UA services in the Session class are accessible as methods on the Session object. The Session object provides a number of methods, for example the Open (String, IUserIdentity) method which creates and activates a session (see Fig. 9). This is responsible for sending and processing the publish requests. The required process in order to establish a session between the OPC UA client and the OPC UA server is shown in Fig. 8. It indicates that the OPC UA client application has to choose the EndpointDescription for use manually or automatically by using Discovery Services.

The Subscription class stores the client state for a subscription with an OPC UA server. It maintains two sets of properties such as the requested values and the current values based on the

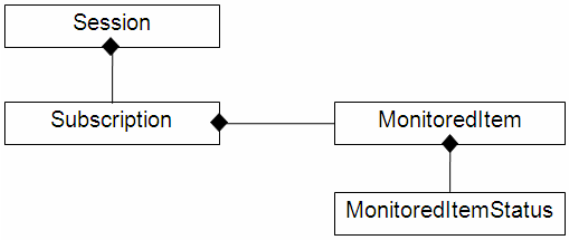

Fig. 7. The classes for the development of OPC UA client applications

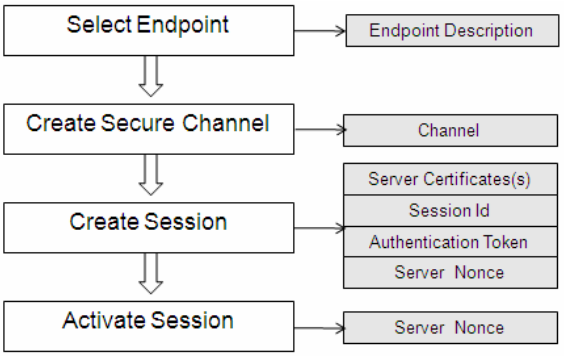

Fig. 8. The required process for establishing a session between the OPC UA client and the OPC UA server

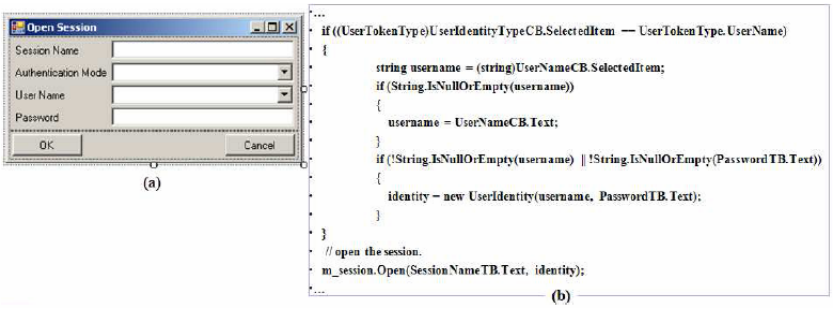

Fig. 9. The Open method for creating a session

revised values provided by the OPC UA server. A Subscription object is designed for batch operations(e.g., the subscription parameters and the monitored items can be updated several times). However, the changes to the subscription on the server do not happen until the method *ApplyChanges*() is called. The parameters for a subscription can be updated and changed as shown in Fig. 10: (a) user interface and (b) code fragment in C# language for updating the changed values to variables. The Subscription maintains a cache of received messages in which the size of the cache can be controlled and set by the MaxMessageCount property, i.e., the maximum number of messages to keep in the internal cache. When a new message is received, the subscription adds it to the cache and removes any extras. The subscription then extracts the notifications and pushes them to the MonitoredItem.

The MonitoredItem class, which stores the client state for a monitored item, belongs to a subscription on an OPC UA server. It maintains the requested values and the current values based on the revised values returned by the server. It also provides a notification event, which can be used by the OPC UA client application to receive events whenever a new notification is received from the OPC UA server. Changes to any of the properties that affect the state of the MonitoredItem on the OPC UA server will be applied when the *ApplyChanges*() method for the subscription is called. A local queue for data exchanges or events received from the OPC UA server is maintained by the MonitoredItem (e.g., the OPC UA client application does not need to explicitly process notification messages and can read the latest value from the MonitoredItem when it is required). The parameters of each monitored item can be changed as well, as shown in Fig. 11.

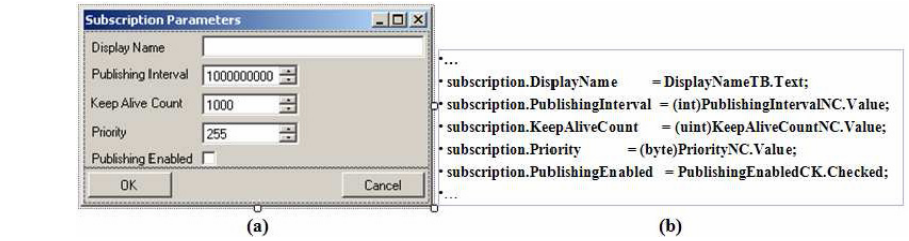

Fig. 10. The interface for updating the subscription parameters

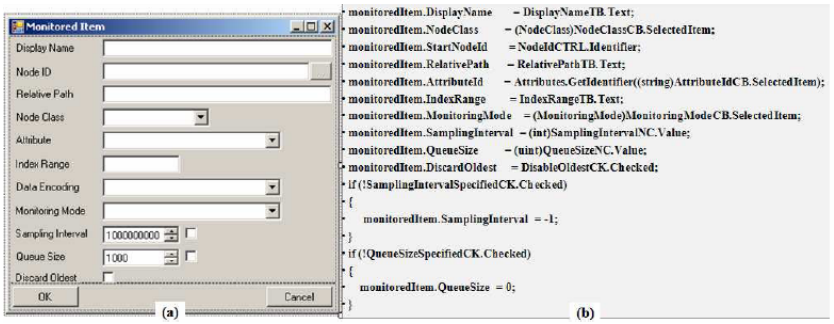

Fig. 11. Interface for changing the parameters of a monitored item

In addition, the size of the local queue can be controlled by the CacheQueueSize property.

5. Potential Areas and Discussion

5.1 Performance

Performance is a key aspect in assessing a technology. In general, it is the transfer performance, the response time, and the ability to handle a large amount of data that decide whether a technology will be used [21]. In this article, the performance measurement covers the round-trips necessary for a Read method call depending on the number of variables (e.g., OPC items [19]) (see Fig. 12). The method was called in a loop for different configurations, for example, XML or binary data encoding, different security modes, etc. The type of data used for the operation Read was a four-byte-integer value and the measurement was performed with server-client applications without application logic just creating return parameters with valid values. The simulation setup was composed of the OPC UA server and the OPC UA client running on Windows XP Service Pack 2 with 2.66 Ghz Intel Pentium IV CPU and 1 Gb PC 3200 DDRAM.

The time taken for operation Read according to the HTTP (remote) or TCP (local) protocol is shown in Fig. 13. It indicates that the performance of the proposed framework used for remote OPC communication is acceptable to many real monitoring and control applications because 5,000 variables in one Read call only take 240 ms. In order to analyze the performance of the OPC UA communication system over the Internet when controlling time critical processes like grinding processes, attention should be focused on the round-trip time of the communication systems because the round-trip time is strictly related to the refresh time of periodical data processes. The proposed framework provides an average round trip enough to support the periodicity

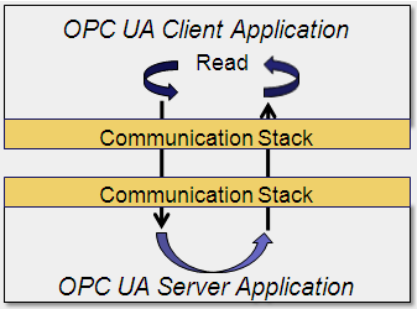

Fig. 12. The application setup for measurement performance

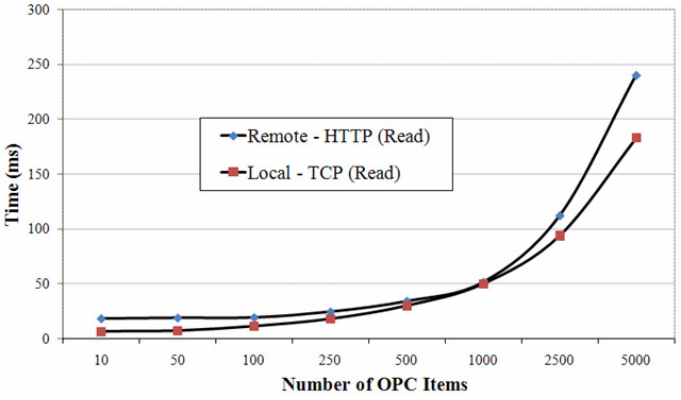

Fig. 13. The time taken for operation Read according to the HTTP (remote) or TCP (local) protocol

for data and event processes around 240 ms for 5,000 items over the HTTP protocol. The initial results in this article can be used to compare with a number from the round-trip time of the different access methods from the Advosol Inc. [2]. The round trip time of the access from a client to a local XML-DA server, an XML-DA Gateway to a local OPC DA server, and a client in USA to a server in Europe, in turn, is 45 ms, 35 ms, and 1.2 s while the round-trip time of the proposed framework is about 18 ms for one item in one Read call.

In addition, to validate performance, the performance comparison of the time taken for operation Read with the fixed conditions such as HTTP, signature and encryption, and 256-bit Basic by either using binary or XML encoding is shown in Fig. 14. It indicates that the number of values transferred with binary encoding is several times higher than with XML encoding. Thus, the overall performance of the system is much improved when using binary data encoding for data changes and event notifications. There is a much bigger overhead when using HTTP protocol with XML encoding instead of binary encoding. It is about 2 times slower for small messages and more than 10 times slower for large messages (see Fig. 14).

In summary, only the initial results have been presented in this article because the proposed framework is being developed and implemented for providing performance validation and a collection of libraries, classes, interfaces, and sample implementations. However, they indicate that the proposed framework has good performance and is acceptable to many monitoring and control applications in the real-world.

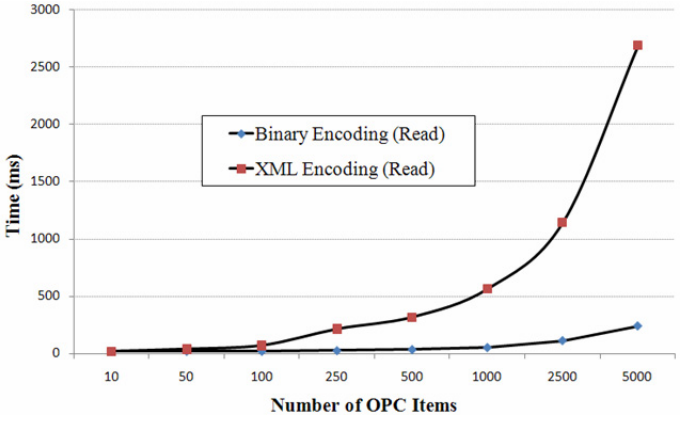

Fig. 14. The performance comparison of the time taken for operation Read under the fixed conditions: HTTP, signature and encryption, 256-bit Basic

5.2 Potential Areas for the Proposed Framework

As the simulation results from the performance tests have indicated the proposed framework has good performance, and the potential areas for the proposed framework include the following:

1. Applications of monitoring and control systems for the purpose of process condition maintenance and monitoring (e.g., automatic data acquisition, condition based maintenance, identifying malfunctioning instruments, monitoring multiple factors, maintaining control loops and more [11]), in process and factory automation where control is distributed and large amounts of distributed data are collected. In addition, the development of industrial applications for integrating plant floor devices and enterprise systems with the control and monitoring part can be satisfied. A modern maintenance system using the proposed framework as an example is shown in Fig. 15. Real-time data is collected and the condition based maintenance system can use this kind of data to make decisions such as the loop and sending out an alarm or event, to re-identify a system, and to monitor a system [24].
2. Business applications such as the Computerized Maintenance Management System (CMMS), Enterprise Resource Planning (ERP), and the Enterprise Asset Management (EAM) systems, which require data updates where the frequency is a few seconds or minutes, etc.
3. Process Analysis applications such as Enterprise Process Historian, Analysis Reporting, Trending, and others. The frequency of data capture for these applications is not nearly as important as the fact that data does not get loss.
4. Remote Monitoring applications such as web-based process visualization, which require very slow data update rates (sometimes every few seconds or minutes) and only limited supervisory control.

5.3 Discussion on Redundancy Strategies

For the applications of the automation industry, redundancy is an important feature for in-

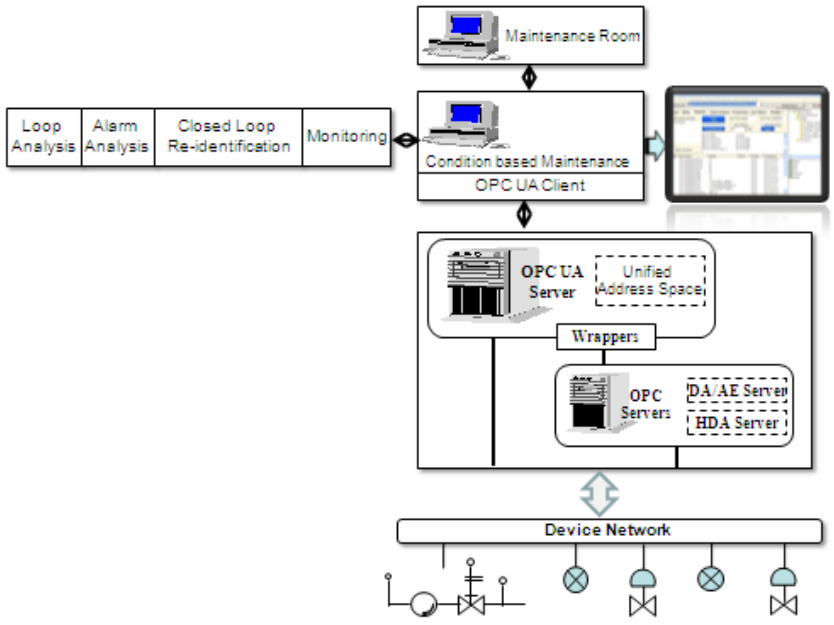

Fig. 15. The concrete architecture of a condition based maintenance system by using the proposed framework

creasing the efficiency and reliability of the system [19]. Redundancy is therefore needed for both cases of the communication link from OPC server and devices, and for the communication between a server and a client. From this point of view, three different redundancy strategies should be considered such as Device Level redundancy, Server Level redundancy, and Client Level redundancy [24]. In a Device Level redundancy strategy, controllers or devices are implemented in a redundant configuration. If the server-device connection fails, then the redundant devices start operating. Hardware vendors guarantee this strategy. Thus, Server Level and Client Level redundancy is considered and discussed in this section. In OPC UA based applications, redundancy is based on the existence of duplicate client or server applications and can be achieved by using particular data structures and services of the OPC UA specifications.

1. **Server Redundancy**. The principle of server redundancy is that there are two servers for providing data to a client. One can be the primary server while the other one is in standby mode, or both are operating at the same time. If the communication between the first server and the client fails, the redundant server supplies the necessary process data. For OPC UA based applications, it must keep data on both servers because the OPC UA server supports historical data. Two modes of server redundancy can be considered and implemented such as Transparent Redundancy and Non-transparent Redundancy [29]. Transparent Redundancy is that server redundancy is handled transparently to the client. The server is responsible to ensure that all information is synchronized between redundant servers (e.g., the redundant servers have to be mirrored or they must have exactly the same data and session information). In contrast, Non-transparent Redundancy requires some actions by the client to continue its tasks when the primary server fails. The concepts of cold, warm, and hot failovers

in Non-transparent Redundancy are defined to deal with various use cases [29, 19].

2. **Client Redundancy**. Client redundancy is needed for environments in which the continuous supervisory of a production process is required. It requires that two clients or applications have to be implemented in a redundant way. If the connection or link to a server or the application itself fails, then the second client starts operating. The backup client monitors the session information of the active client in the server's address space [19]. When the active client fails and the status of the session changes in the address space, the backup client would receive the appropriate notification from the server. The backup client then uses the TransferSubscription service to get all running subscriptions from the active client by requesting the server to transfer the subscriptions and sessions (e.g., the TransferSubscription service is used to transfer a subscription and its Monitored Items from one session to another [21]). If the backup client meets the correct security requirements, the subscriptions will successfully transfer. For example, a redundant approach uses at least a heartbeat signal in which two clients are active and inform each other about its current health state, or the operating client activates the backup client if a standby mode is preferred [24].

6. Concluding Remarks and Future Works

In this article, the current technological state of OPC standards and the current problems of these standards were provided and discussed. An OPC UA client-server framework for the development of monitoring and control applications in the automation industry was developed and presented by using the Visual Studio 2008 and OPC UA SDK. The proposed framework is considered as an integration solution for the monitoring and control of devices on the plant floor or shop floors, and condition based maintenance systems according to the data and events received from the devices. This framework provides a collection of libraries, classes, reference interfaces and sample implementations that make it easy for developers and programmers to create and implement their OPC components and related applications for the terms of monitoring and control. Moreover, exposing information in much more semantics based on the OPC UA Information Model allows OPC clients to process highly sophisticated tasks in industrial and informatics systems. Such OPC clients can access device information, device data, and events, which are provided by different and vendor-specific OPC UA servers in the same manner.

The proposed framework in turn minimizes the efforts of developers in learning new techniques and allows system architects and designers to perform dependency analysis on the development of industrial automation applications. The potential areas of the proposed framework and the redundancy strategies to increase the efficiency and reliability of the system were also considered and provided. The initial simulation results have indicated the performance of the proposed framework is sufficient and acceptable to many industrial applications in the domains of the monitoring and control system.

A challenging task for future works is to actually implement, evaluate, and refine the proposed framework for applying to real-world applications. In addition, non-functional issues like performance, scalability, communication overhead, and security need to be monitored and deeply investigated. The redundancy strategies, as adequately aforementioned, will be developed and implemented by using special data structures and services that were mentioned in the OPC UA specifications. These strategies will be investigated carefully before deploying the proposed

framework to real applications.

REFERENCE

[1] Advosol, Inc., http://www.advosol.us/

[2] Advosol, Inc.: "*The Advosol Benchmarks*," http://www.advosol.us/t-WhitePaperXMLDANET.aspx

[3] J. Bean: "*SOA and Web Services Interface Design - Principles, Techniques, and Standards*." Morgan Kaufmann OMG Press (2010)

[4] COM: *Component Object Model Technologies*, http://www.microsoft.com/com/default.mspx

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

[6] T. Cucinotta, A. Mancina, G.F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, F. Rusina: "*A Real-Time Service-Oriented Architecture for Industrial Automation.*" IEEE Transactions on Industrial Informatics, vol. 5, no. 3, 2009, pp. 267-277.

[7] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana: "*WSDL - Web Services Description Language, Version 1.1,*" 2001, http://www.w3.org/TR/wsdl

[8] L.M.S. de Souza, P. Spiess, D. Guinard, M. Kohler, S. Karnouskos, D. Savio: "*SOCRADES - A Web Service Based Shop Floor Integration Infrastructure.*" In: C. Floerkemeier, M. Langheinrich, E. Fleisch, F. Mattern, S.E. Sarma (eds) IOT 2008. Lecture Notes in Computer Science, vol. 4952, 2008, pp. 50-67. Springer, Heidelberg.

[9] D. Dzung, M. Naedele, T.P. Von Hoff, M. Crevatin: "*Security for Industrial Communication Systems.*" Proceedings of IEEE, vol. 93, no. 6, 2005, pp. 1152-1177.

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

[11] F. Gord: "*CMMS: Integrating Real-Time Information for Condition-Based Maintenance.*" Matrikon, Inc., 2005, www.MatrikonOPC.com/tutorial

[12] K.H. Han, S. Kim, Y.J. Kim, J.H. Kim: "*Internet Control Architecture for Internet-based Personal Robot.*" Autonomous Robots, vol. 10, 2001, pp. 135-147.

[13] F. Iwanitz, J. Lange: OPC: "*OPC: Fundamentals, Implementation, and Application."* Huthig Verlag Heidelberg, 3rd rev. Ed., 2006.

[14] F. Jammes, H. Smit: "*Service-Oriented Paradigms in Industrial Automation.*" IEEE Transactions on Industrial Informatics, vol. 1, no. 1, 2005, pp. 62-70.

[15] Z. Jia, X. Li: "*OPC-based Architecture of Embedded Web Server."* In: Z. Wu, C. Chen, M. Guo, and J. Bu (eds.) ICESS 2004. Lecture Notes in Computer Science, vol. 3605, 2005, pp. 362-367. Springer, Heidelberg.

[16] S. Karnouskos, O. Baecker, L.M.S de Souza, P. Spiess: "*Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure*." In Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation, 2007, pp. 1-8. IEEE Press, Los Alamitos.

[17] M. Khalgui, X. Rebeuf, F. Zampognaro: "*Adaptable OPC-XML Contracts Taking into Account Network Traffic.*" In Proceedings of the 10th IEEE Conference on Emerging Technologies and Factory Automation, 2005, pp. 31-38. IEEE Press, Los Alamitos.

[18] R. Kondor: "*OPC, XML, .NET and What it all Means Down on the Factory Floor.*" Industrial Ethernet Book, 2007, Issue 34:38, http://ethernet.industrialnetworking.com/articles/articles.asp

[19] W. Mahnke, S.H. Leitner, M. Damm: "*OPC Unified Architecture.*" Springer, Heidelberg, 2009.

[20] MatrikonOPC, 2010, http://www.matrikonopc.com/

[21] J. Lange, F. Iwanitz, T.J. Burke: "*OPC - From Data Access to Unified Architecture.*" VDE Verlag GMBH, 4th rev. Ed., 2010.

[22] W. Roshen: "*SOA Based Enterprise Integration – A step-by-step Guide to Services Based Application*

Integration." McGraw-Hill Companies, 2009.
[23] C. Sahin, D. Bolat: "*Development of Remote Control and Monitoring of Web- Based Distributed OPC system.*" Computer Standard & Interfaces, vol. 31, 2009, pp. 984-993.
[24] M.H. Schwarz, J. Boercsoek: "*Advances of OPC Client Server Architectures for Maintenance Strategies - a Research and Development Area not only for Industries.*" WSEAS Transactions on Systems and Control, vol. 3, no. 3, 2008, pp. 195-207.
[25] M.P. Singh, M.N. Huhns: "*Service-Oriented Computing: Semantics, Processes, and Agents*." Wiley & Sons, Chichester, 2005.
[26] V.V. Tan, M.-J. Yi: "*Design Issues and Approach to Internet Based Monitoring and Control Systems*." In: Garcia-Pedrajas, N. et al. (eds.) IEA/AIE 2010. Lecture Notes in Artificial Intelligence, Part I, vol. 6096, 2010, pp. 478-488. Springer, Heidelberg.
[27] V.V. Tan, M.-J. Yi: "*OPC UA Based Information Modeling for Distributed Industrial Systems."* In: Huang, D.-S. et al. (eds.) ICIC 2010. Lecture Notes in Computer Science, vol. 6215, 2010, pp. 531-539. Springer, Heidelberg.
[28] The OPC Foundation, http://www.opcfoundation.org/
[29] The OPC Foundation: "*The OPC Unified Architecture Specifications: Parts 1-11, Version 1.xx,*" 2009, http://www.opcfoundation.org/Downloads.aspx
[30] The OPC Foundation: "*The OPC Alarms and Events Specification, Version 1.0,*" 2002. http://www.opcfoundation.org/Downloads.aspx
[31] The OPC Foundation: "*The OPC Historical Data Access Specification, Version 1.0,*" 2003. http://www.opcfoundation.org/Downloads.aspx
[32] The OPC Foundation: "*The OPC XML-Data Access Specification, Version 1.01,*" 2004. http://www.opcfoundation.org/Downloads.aspx
[33] The OPC Foundation: "*The OPC Data Access Specification, Version 3.0,*" 2004. http://opcfoundation.org/Downloads.aspx
[34] N.M. Torrisi, J.F.G. Oliveira: "*Remote Control of CNC Machines Using the Cyber OPCCyberOPC Communication System over Public Networks.*" International Journal of Advanced Manufacturing Technology, vol. 39, no. (5-6), 2008, pp. 570-577.
[35] K. Usami, S.-I. Sunaga, H. Wada: "*A Prototype Embedded XML-DA Server and its Evaluations*." In Proceedings of the SICE-ICASE International Joint Conference, 2006, pp. 4331-4336.
[36] M. Venzke, C. Weyer, V. Turau: "*Application Specific vs. Standard Web Service Interfaces for the Vertical Integration of Fieldbus Systems.*" In Proceedings of the 3rd International Workshop on Intelligent Solutions in Embedded Systems, 2005, pp. 153-162. IEEE Press, Los Alamitos.
[37] S. Wang, Z. Xu, J. Cao, J. Zhang: "*A Middleware for Web Service-Enabled Integration and Interoperation of Intelligent Building Systems.*" Automation in Construction, vol. 16, 2007, pp. 112-121.
[38] S.H. Yang, X. Chen, J.L. Alty: "*Design Issues and Implementation of Internet-based Process Control Systems.*" Control Engineering Practice, vol. 11, 2003, pp. 709-720.
[39] F. Zhao, J. Chen, G. Dong, L. Guo: "*SOA-Based Remote Condition Monitoring and Fault Diagnosis System.*" International Journal of Advanced Manufacturing Technology, vol. 46, 2010, pp. 1191-1200.

Vu Van Tan

He was born in Haiduong Province, Vietnam, in 1981. He received the engineer degree in Information Technology from the Hanoi University of Technology, Vietnam, in 2004. He also received the PhD degree in Information Technology from the University of Ulsan, Republic of Korea, in 2010. He has worked as a design and analysis engineer in KhaiTri Software Company, Vietnam, for one year (June 2004 – August 2005). He is currently a postdoctoral researcher at the Network Based Automation Research Center (NARC), University of Ulsan. His main interests are software engineering, software for automation systems, internet technologies for industrial automation systems, agent technologies for process monitoring and control, and real-time communication systems.

Myeong-Jae Yi

He received the BS degree in Computer Science from the Seoul National University, Republic of Korea, in 1987. He also received the MS and PhD degrees in Computer Science from the Seoul National University in 1989 and 1995, respectively. He was a part-time lecturer at the Department of Computer Science of the Seoul National University and the Sookmyung Women's University from 1991 to 1996. He is currently a professor in the School of Computer Engineering and Information Technology, University of Ulsan, Republic of Korea. Professor Yi is also a deputy-director of NARC (Network based Automation Research Center) at the University of Ulsan and is a member of KIISE and KIPS. His main interests are software engineering, software for automation systems, Internet technologies for industrial automation systems, mobile agent, and E-commerce.