A FUNCTIONAL FRAMEWORK FOR THE WEB-BASED DISTRIBUTED CONTROL SYSTEMS

Changho Lee [*] **Yoo-Sung Kim** [**] **Jaehyun Park** [***]

[*] *Dept. of Automation Engineering,*
Inha University, Inchon, Korea, Email:chlee@resl.inha.ac.kr
[**] *Dept. of Communication and Information Engineering,*
Inha University, Inchon, Korea, Email:yskim@inha.ac.kr
[***] *Dept. of Communication and Information Engineering,*
Inha University, Inchon, Korea, Email:jhyun@inha.ac.kr

Abstract: Emerging IT technologies, specially Internet communication and web-based applications are adopted to the modern distributed control systems. This paper defines a functional framework for the web-based application of a distributed control system connected by Internet. XML(eXtensible Markup Language) is used for representing a control system and control devices. These IT technologies makes a distributed control system more flexible and scalable than existing distributed systems with the standard RMI protocols such as CORBA or DCOM.

Keywords: Open architecture, distributed control system, SOAP, XML, real-time control.

1. INTRODUCTION

Modern Internet and information technology(IT) enables control systems to utilize the distributed environment connected by high speed networks, such as field-buses, Ethernet, ATM, and wireless networks. With adopting these information technologies in WAN(Wide Area Network), the geometric range for a distributed control system becomes wider and wider, and in turn, a new software framework is required for this new environment(Lee and Park, 2000).

A software for the Internet-based distributed control systems is usually designed using the standard software components that operate on the top of the industry standard middlewares such as CORBA (Common Object Request Broker Architecture) from OMG (Object Management Group) (OMG, 1996; Orfali *et al.*, 1998) and COM (Component Object Model) from Microsoft(Microsoft, 1996; Caron, 2000). With using these kinds of middlewares that support the network-based software components, a distributed control system becomes more flexible and manageable.

However, because these middlewares are developed primarily for the general Internet-based applications, they have inherently several problems to be directly used for the control systems(Grimes, 1997; Daniel and Vallee, 1999).

- Since two major middleware standards, DCOM and CORBA, are not compatible with each other, it is hard to maintain the interoperability between them(Chang *et al.*, 2000).
- Since the network messages for these middlewares hardly pass through the security barrier like a firewall, they are not likely used in the distributed control systems across the WAN boundaries.
- Since these middlewares are primarily developed for the general-purpose operating systems such as Microsoft Windows or Unix, it is difficult to apply them to the embedded control systems directly(Chen *et al.*, 1999; Chisholm, 1997).

⋆ The work reported in this paper is partially supported by Korea G7 project(grant number G7/5-3-1-4).

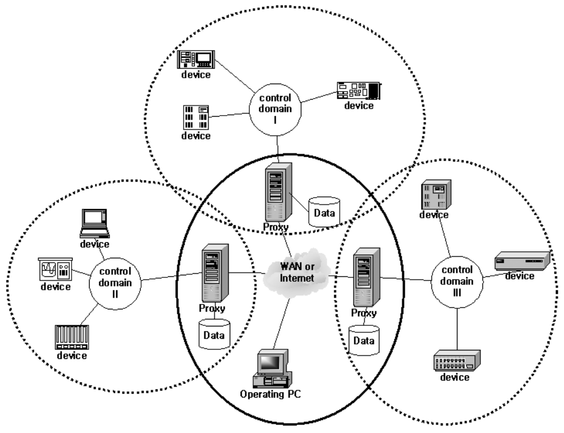


Fig. 1. Global Control Domain

To overcome these difficulties in using CORBA or DCOM, a new protocol, SOAP (Simple Object Access Protocol), is proposed as a RMI(Remote Method Invocation) for the Web-based applications. Since SOAP uses HTTP(Hyper Text Transfer Protocol), it is suitable for designing a control systems in WAN environment. However, the current version of SOAP standard defines only a simple protocol format for a remote method invocation, a SOAP message structure should be defined in detail to use SOAP for a distributed control system. This paper defines the SOAP message structure for a distributed control system for the remote monitoring and control.

This paper is organized as follows. Section 2 introduces the background technologies that are needed to design the control system. Section 3 proposes a control domain and SOAP message structure. Section 4 demonstrates the implementation example.

2. BACKGROUND TECHNOLOGY

2.1 *eXtensible Markup Language(XML)*

XML, a standard protocol defined by W3 organization, is a meta-language to describe other languages(Bray *et al.*, 2000). Since the underlying philosophy of XML aims flexibility and portability, it is possible to design a platform-independent RMI using XML. XML can be considered as two kinds of objects: a document and data. XML as a document, is used to make a document and define tags and technologies. XML as a data, is considered as a transfer syntax as well as data types(Box, 2000*a*). In addition, hierarchical data structure of XML is very useful to convey the internal data structure between the distributed control systems. Recently, to utilize XML data more efficiently, XML Information Set(XML InfoSet) is proposed. With XML InfoSet, network traffic can be reduced for the real-time communication.

2.2 *Simple Object Access Protocol*

SOAP (Simple Object Access Protocol) is a HTTP-based RMI(Remote Method Invocation) protocol and has a message structure defined by XML standard(Sturm, 2000). Since only a web server module is necessary to use SOAP, ranges of embedded control systems are able to maintain interoperability with other platforms using SOAP. The packet format of SOAP is similar to that of existing ORPC(Object Remote Procedure Call) that is used in DCOM or CORBA IIOP/GIOP. This similarity makes it easy to convert most of the existing objects and methods written in DCOM and CORBA to SOAP-based objects and methods. For the object references and method requests, SOAP uses URL(Uniform Resource Locator) and URI(Uniform Resource Identifier)(Box, 2000*b*). Even SOAP has the same function as ORPC, SOAP has following advantages over other RMI methods: First, since SOAP is a plain text, it is easy to bind various protocols. Second, while IIOP(Internet Inter-ORB Protocol) of CORBA and Java RMI require very bulky infrastructure for a remote object reference or a garbage collection, SOAP doesn't require it.

3. WEB-BASED FRAMEWORK FOR CONTROL DOMAIN

This section defines the SOAP message format and XML document for the application software for a distributed control system.

A large distributed control system, defined as a control domain, consists of several sub-domains that include multiple devices as shown in Figure 1. Internetworking between control domains is gated by a *proxy server*. The proxy server between control sub-domains gives several merits. First, the proxy server can moderate the network traffic, that is very important issue for the real-time operation of control systems. If a real-time control system is open to the public Internet, it experiences high network load as well as attacking from outside. For the real-time operation, the network between control devices should be isolated from the public Internet. This isolation is easily achieved by using the proxy server, with which each control devices exchange their data with each other without being interrupted from the outside traffic. Second, the proxy server can maintain a high level of security. Real-time control system can be efficiently protected from the active attacks such as DOS(Denial of Service). Last but important point is to compensate the inherit stateless operation of SOAP. Without any state information, maintaining the control information efficiently is almost impossible and large network load is usually injected. But with a proxy server, state-oriented communication can be efficiently managed even SOAP itself doesn't support it(Microsoft, 2000).

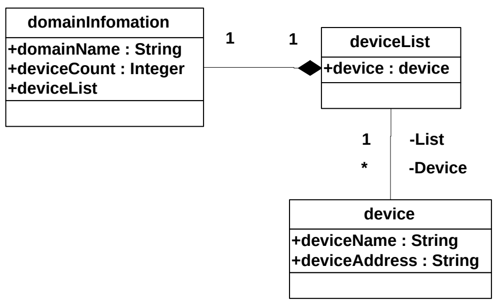


Fig. 2. Control Domain DDL

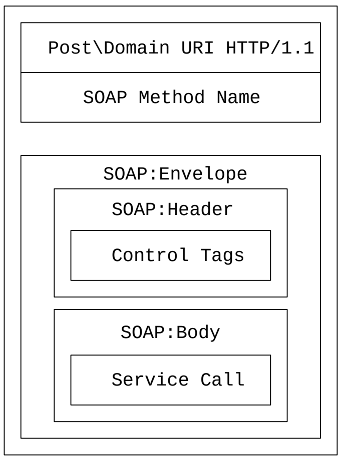


Fig. 3. SOAP Message Format

3.1 *Control Domain Abstraction*

To use SOAP for a control domain, a control domain itself as well as the services provided by the control devices should be well defined in XML format. These definitions are called *Domain Description Language (DDL)* and *Service Description Language (SDL)* respectively, and they reside in the proxy server that represents a particular control domain.

Figure 2 shows the proposed XML schema of DDL for a control domain. A DDL consists of three major elements: `domainName`, `deviceCount`, and `deviceList`. `domainName` represents the name of current control domain. `deviceCount` means the number of devices registered in a control domain. The devices within a control domain is described in the `deviceList` that has two elements: `deviceName` and `deviceAddress`. With `deviceName` element, user can fetch its SDL (deviceName.sdl) from a proxy server, and, in turn, get services and parameters. A `deviceAddress` element is used to inform the proxy server of the physical address of a device. In Internet, `deviceAddress` is represented by IP Address or hostname.

3.2 *SOAP Message for Control Domain*

This section proposes a SOAP message format for a control domain whose abstraction is defined in the

```
<!ELEMENT SOAP-ENV:Envelope (SOAP-ENV:Header)>
<!ATTLIST SOAP-ENV:Envelope     xmlns:xsi   CDATA #REQUIRED
               xmlns:SOAP-ENV  CDATA #REQUIRED >

<!ELEMENT SOAP-ENV:Header      (trans:transfer , const:constraint )>
<!ATTLIST SOAP-ENV:Header   xsi:type CDATA #REQUIRED >
<!ELEMENT trans:transfer    (trans:to , trans:from )>
<!ATTLIST trans:transfer    xmlns:trans CDATA #REQUIRED
            xmlns:agr   CDATA #REQUIRED
            SOAP-ENV:mustUnderstand CDATA #REQUIRED>
<!ELEMENT trans:to (trans:address )>
<!ELEMENT trans:address EMPTY>
<!ATTLIST trans:address     xsi:type    CDATA #IMPLIED >
<!ELEMENT trans:from    (trans:address )>
<!ELEMENT const:constraint  (const:timestamp ,  const:timeout  )>
<!ATTLIST const:properties  xmlns:const CDATA #REQUIRED
                SOAP-ENV:mustUnderstand CDATA #REQUIRED>
<!ELEMENT const:timestamp       (#PCDATA )>
<!ELEMENT const:timeout     (#PCDATA )>
```

Fig. 4. SOAP Control Tags

previous section. Since current SOAP standard defines only the minimal set of SOAP header, the detail structure of header and body for a specific domain should be defined. Figure 3 shows the SOAP message structure for the control domain proposed in this paper.

3.2.1. *Control Tags* Figure 4 shows the internal structure of SOAP header used in this paper. As described above, the control domain defined in this paper uses a proxy server, all information related to message transfer should be registered in the proxy server. This information includes address information and timing constraint required for the real-time control. To provide information of control devices for the proxy server, *control tag* is defined in SOAP header, that includes two tags.

`transfer tag`
In the control domain proposed in this paper, since control devices are isolated from the external domains as described above, user should provide all of the information to drive a device in order that a proxy server could interface with control devices instead of clients. For this purpose, a mandatory element, `transfer tag`, is defined. `transfer tag` includes the location information of SOAP message source and destination.

`constraint tag`
A constraint tag contains the timing information to decide whether a tranferred message meets the real-time constraints or not. Sender records the timestamp in a `timestamp` element, and timeout value in a `timeout` element. If the timeout value is expired, a message is considered as an invalid message.

3.2.2. *Service Call* For the security reason, a user is required to be authorized for accessing the control system. A proxy server can authorize user to access the control domain using an industry standard authentication mechanism. After authentication process, users can obtain the domain information written in DDL and SDL from the proxy server. With the information obtained, the service request and the response

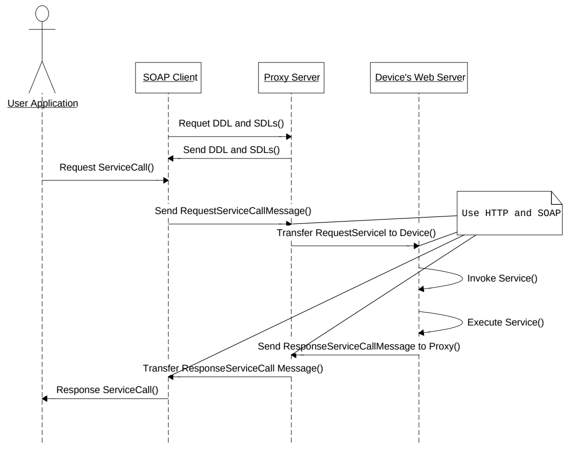


Fig. 5. SOAP-Compliant Client Processing SOAP

```
<?xml version="1.0" ?> <serviceDescription
    name="SoapService"
    xmlns="urn:schemas-xmlsoap-org:sdl.2000-01-25"
    xmlns:dt="http://www.w3.org/1999/XMLSchema"
    xmlns:ss ="#SimpleStuff">
    <import namespace="#SimpleStuff"
        location ="#SimpleStuff"/>
    <soap xmlns="urn:schemas-xmlsoap-org:soap-sdl-2000-01-25">
        <service>
            <requestResponse name="Method">
                <request ref="ss:MethodReq" />
                <response ref="ss:MethodRes" />
            </requestResponse>
       </service>
    </soap>
    <ss:schema  id="SimpleStuff"
            targetNamespace="http://server-url/services.xml"
            xmlns:dt="http://www.w3.org/1999/XMLSchema"
                xmlns="http://www.w3.org/1999/XMLSchema">
        <element name="MethodReq"/>
            <type>
                <element name="Input" type="dt:int" />
            </type>
        <element name="MethodRes">
            <type>
                <element name="Output" type="dt:int" />
            </type>
        </element>
    </ss:schema>
</serviceDescription>
```

Fig. 6. Service Description Language Example

messages are exchanged as shown in Figure 5. The internal structure of the request and response message written in SDL is shown in Figure 6.

3.3 *Registration and Management of Control Devices*

Since DDL locates in the proxy server, each device must register its services to the proxy server. A control device can register its IP address and SDL to the proxy server through a web service defined in Figure 9. As shown in Figure 7, when the proxy server receives a registration message from a device, it determines whether a device is already registered in the `deviceList` or not. If a device is not registered in the list, the proxy server adds a new device into the list. Otherwise, it just updates the information of the device in the list.

After the registration process completes, the proxy server uses the `CheckDevice` method to check the device is removed from a domain. If the device returns failure code or doesn't respond to the `CheckDevice`

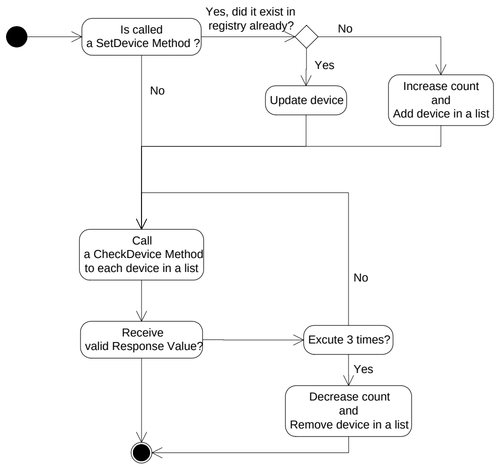
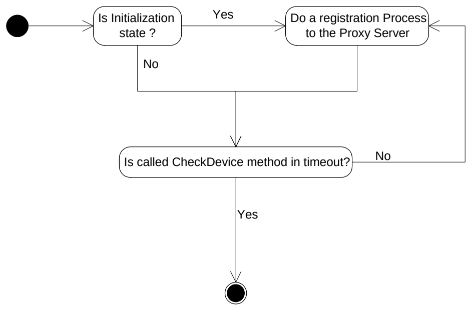


Fig. 7. Registration and Management Process State Diagram at a Proxy Server



Is Initialization
state ?
Yes
Do a registration Process
to the Proxy Server
No
Is called CheckDevice method in timeout?
No
Yes

Fig. 8. Registration Process State Diagram at a Control Device

```
ProtoType : boolean SetRegister(string, string, string);
            int     CheckDevice(int);

<ss:schema  id="RegisterStuff"
        targetNamespace="http://server-url/services.xml"
        xmlns:dt="http://www.w3.org/1999/XMLSchema"
            xmlns="http://www.w3.org/1999/XMLSchema">
    <element name="SetRegisterReq"/>
        <type>
            <element name="Addr" type="dt:string" />
            <element name="DomainName" type="dt:string" />
            <element name="SDL" type="dt:string" />
        </type>
    <element name="SetRegisterRes">
        <type>
            <element name="RetureCode" type="dt:boolean" />
        </type>
    </element>
</ss:schema>

<ss1:schema  id="HeartbeatStuff"
        targetNamespace="http://server-url/services.xml"
        xmlns:dt="http://www.w3.org/1999/XMLSchema"
            xmlns="http://www.w3.org/1999/XMLSchema">
    <element name="CheckDeviceReq"/>
        <type>
            <element name="ConnectionReference" type="dt:int" />
        </type>
    <element name="CheckDeviceRes">
        <type>
            <element name="ReferenceValue" type="dt:int" />
        </type>
    </element>
</ss1:schema>
```

Fig. 9. SDL of Registration and Management Methods

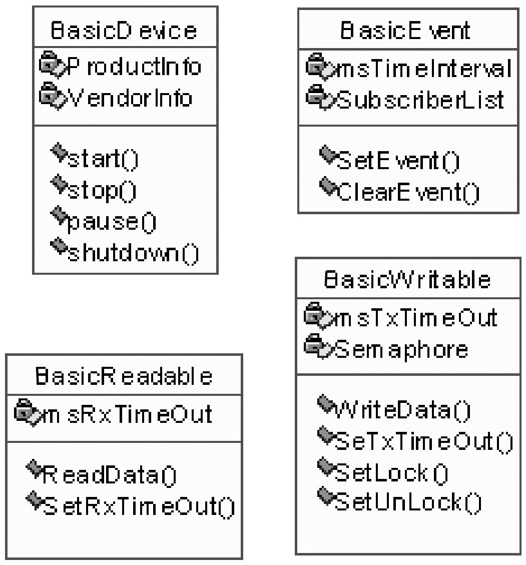


Fig. 10. Basic Services

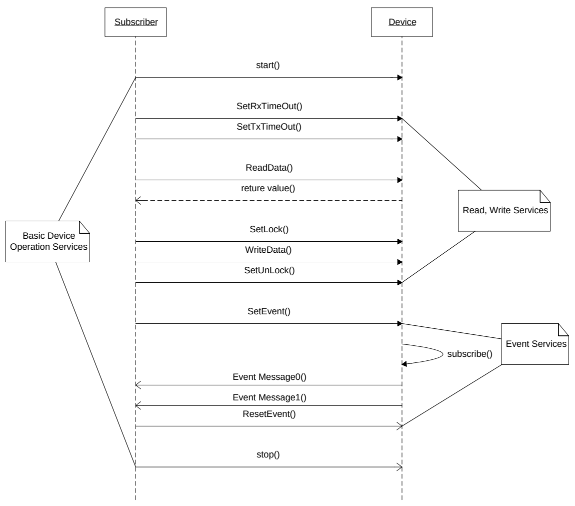
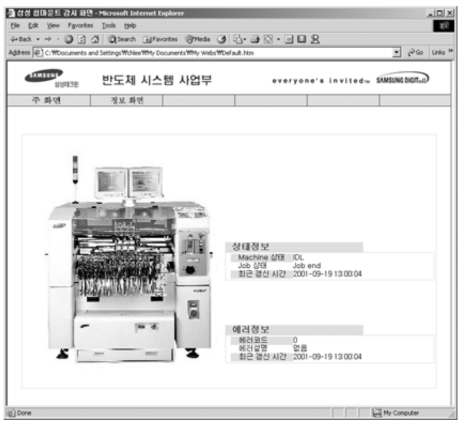


Fig. 11. Sequence Diagram of Services

message, the proxy server assumes that it is not in a domain and removes it from the list. On the contrary, if there is no `CheckDevice` message from the proxy server during a certain period, a control device requests a registration process as shown in Figure 8.

3.4 *Basic Control Services*

To adopt the SOAP protocol to the control systems, basic control services should be defined. Figure 10 shows four service categories: `BasicDevice`, `BasicReadable`, `BasicWritable`, and `BasicEvent`. `BasicDeive` includes the fundamental services such as device initialization and operation control. `BasicReadable` and `BasicWritable` provide the services neccessary for reading and writing data from/to the control systems. Last service group, `BasicEvent`, includes the services used for


삼성 칩마운트 감시 화면 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Documents and Settings\chlee\My Documents\My Webs\Default.htm
Go Links
SAMSUNG 삼성테크윈
반도체 시스템 사업부
everyone's invited SAMSUNG DIGITall
주 화면
정보 화면
Done
My Computer

상태정보

Machine 상태	IDL
Job 상태	Job end
최근 갱신 시간	2001-09-19 13:00:04

에러정보

에러코드	0
에러설명	없음
최근 갱신 시간	2001-09-19 13:00:04

Fig. 12. Implemented Device's Methods

the event handling that plays an important role for the control system monitoring. The interface information of the basic control services written in SDL(Service Description Language) is exported to the exteranl users. To use these services within a control domain, the procedural sequence is shown in the sequence diagram of Figure 11.

4. IMPLEMENTATION

Using the Web-based frameworks proposed in this paper, a demonstration system is implemented. Target application domain is a distributed controller for a semiconductor manufacturing machine, Samsung's commercial chip mounter (Samsung CP-40). To demonstrate the O/S-independent characteristics of the protocol, the devices in the domain are designed on two different operating systems: Microsoft Windows and embedded Linux.

SEMI (Semiconductor Equipment and Material International Inc.) introduced the SECS (SEMI Equipment Communication Standard) protocol to reduce the cost and to improve communication ability between hosts and equipments. This protocol consists of SECS-I for the message communication and SECS-II for the message format. SECS-I uses point-to-point protocol such as RS-232. Recently, HSMS (High-Speed SECS Service) based on TCP/IP is used widely. GEM(Generic Model for Communications and Control of SEMI Equipment) is the protocol that is used between semiconductor equipment and host computer on the top of SECS-I, II and HSMS. In this paper, the proposed SOAP message is used to integrate GEM-based semiconductor manufacturing machines.

Implemented system consists of a proxy server and several devices. The proxy server is implemented on a Microsoft Windows 2000 platform and IIS5.0. It stores and supplies the device's information in DDL and exchange various SOAP messages. As an authentication protocol, the *Kerberos* authentication protocol is used, that is also independent of platforms. Various target control devices are implemented with

Microsoft Windows family(2000, CE) and embedded Linux. Figure 12 shows a demonstration system.

5. CONCLUSIONS

Emerging IT technology, specially Internet and web-based communication, is likely to be adopted to distributed control systems recently. This paper defines web-based control domain connected by Internet, and proposes a functional framework for it. Control domains defined in this paper are connected with each other via a proxy server. Under the control of proxy server, the real-time operation and the security of control domain can be maintained systematically. For openness, a control domain and control devices are abstracted in XML, and SOAP is used for RMI(remote method invocation). Based on SOAP, a set of services for the control and monitoring operation are defined. With adopting these IT technologies, a distributed control system becomes more flexible and scalable than existing distributed systems even they use a standard RMI method such as CORBA or DCOM. At last, the proposed protocols can be easily implemented on the control devices regardless of operating systems, total costs and developing time can be reduced. One drawback of the proposed protocol is high overhead caused by raw XML data and HTTP protocols. These problems can be, however, reduced with XML Infoset mechanism and other methods.

6. REFERENCES

Box, Don (2000*a*). *Essential XML beyond Markup*. Addison Wesley.

Box, Don (2000*b*). HTTP + XML=SOAP. *MSDN Magazine* **15**(3), 67–81.

Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen and Eve Maler (2000). Extensible markup language (xml) 1.0 (second edition). Technical report. W3C.

Caron, Rob (2000). Web services and the soap for visual studio 6.0. *MSDN Magazine* **15**(8), 62–73.

Chang, E., D Annal and F Grunta (2000). A large scale distributed object architecture - corba and com for real time systems. *Third IEEE International Symposium* pp. 338–341.

Chen, Deji, Aloysius Mok and Mark Nixon (1999). Real-time support in com. In: *Proceedings of the 32nd Hawaii International Conference on Systems Sciences*. Hawaii, USA.

Chisholm, Al (1997). OPC Overview. Technical report. OPC Foundation.

Daniel, Jerome and Bruno Traversonand Vincent Vallee (1999). Active com : An inter-working framework for corba and dcom. *Proceedings of the International Symposium, Distributed Objects and Applications*.

Grimes, Rechard (1997). *Professional DCOM Programming*. Wrox.

Lee, Wongoo and Jaehyun Park (2000). A design of infrastructure for control/monitoring system in the distributed computing environment. In: *Procceedings of 15th Korea Automation Control Conference*. Yongin, Korea.

Microsoft (1996). Dcom technology overview. Technical report. Microsoft Corporation.

Microsoft (2000). Universal plug and play device architecture. Technical report. Microsoft Corporation.

OMG (1996). Corba services:common object services specification. Technical report. OMG,Inc.

Orfali, Robert, Dan Harkey and Jeri Edwards (1998). *Instant CORBA*. John Wiley & Sons.

Sturm, Jake (2000). *Developing XML Solutions*. Microsoft Press.