# Component Programming for Power System Software Development

**Yuan Liao* and Jong-Beom Lee****

**Abstract** - This paper illustrates applications of the Component Object Model (COM) for power system software developments. As an integral part of Microsoft's component services, COM has gained increased popularity for developing enterprise applications. This paper illustrates the concepts of COM and the latest developments as well as the available tools for developing COM components. Possible COM applications for developing power system software, such as sparse matrix manipulation, fault location, genetic algorithm applications, and so on. are presented. Advantages and promises brought about by COM are manifested through these examples.

**Keywords**: Component Object Model (COM), software architecture, structural programming, object-oriented programming

## 1. Introduction

Applications of computer tools in electric power systems range from substation automation, monitoring, and control to large scale power system planning and optimization. Enormous effort has been made to develop software for various purposes and future efforts will develop larger, faster, and better software. Certainly, good algorithms must be developed before high quality software can be implemented. For example, to design a power system planning software package, the algorithms for unit commitment, economic dispatch, power market design, and so on. must be appropriately schemed. After the mathematical algorithms for individual functional modules are specified and documented, the next step is to propose a good software architecture. The design of the architecture should allow the software to be easily maintained, expanded, and upgraded. This paper applies component based programming for easier maintenance and reuse.

To implement the mathematical algorithms using computer languages, various programming paradigms are available. Two basic alternatives, namely structural programming and object-oriented programming (OOP), may be used.

In structural programming, a complex system is divided into subtasks and sub-modules that are easier to control and manage [1]. However, structural programming is a data and procedure oriented design method, where the data and procedure are independent entities. Programmers have to bear in mind the data format when dealing with a procedure that hinders code reuse.

* Corporate Research Center, ABB, Raleigh, USA(yuan.liao@us.abb.com).
** Dept. of Electrical and Electronic, & Information Engineering, Wonkwang University, Korea (ipower@wonkwang.ac.kr).
Received November 7, 2002 ; Accepted November 28, 2002

To solve this problem, OOP assimilates all the benefits of the structural programming and provides encapsulation. In OOP, the data and the operations on the data (also called methods) are considered an inseparable part of using data abstraction and hiding. The object and the method are abstracted into a new data type called class. The relationship between different classes and class reuse are fully considered. OOP is promising for generating reusable and maintainable codes. The software systems built on an object structure are more robust and expandable. The OOP paradigm is embodied by OOP languages. Examples of OOP languages include the LISP family, Simula, Smalltalk, and the C++ family.

OOP techniques make software reuse and maintenance easier. OPP alone, however, is insufficient to solve the problems facing software engineers. For example, when one module of a software package is updated, all the other modules need to be recompiled. This could be a disaster for a largescale package consisting of millions of lines of codes. Another example is the difficulty of reusing a module developed in Visual Basic (VB) in a software package developed in Visual C++ (VC). When a circuit engineer designs a circuit, he simply picks up the electronic component needed and assembles them together. This kind of reuse and flexibility is desired by software engineers, too. The Component Object Model (COM) created by Microsoft Corporation provides a promising way for realizing such a purpose [2–6].

COM has just caught the attention of the power system community. Dugan and McDermott[2] introduced the potential applications of COM for packaging power system analysis tools for better access and easier integration. Attempts are made to establish standard interfaces for building power system software by using the distribution system

simulation software. The implemented system is intended to be more scalable, extendable, and upgradable.

This paper introduces COM and illustrates its possible applications to power system software developments.

## 2. Introduction to COM

This section illustrates the basic concepts and advantages of COM, the available tools for COM programming, and how to deploy the COM application software.

### 2.1 Concepts

Similar to the Common Object Request Broker Architecture (CORBA) developed by the Object Management Group, COM is another object model for component-based programming provided by Microsoft Corporation. COM has been rapidly accepted by the programming community in the past several years [4].

COM is one of Microsoft's component services. Together with other component services including Distributed COM, Microsoft Transaction Server (MTS), Microsoft Internet Information Server (IIS), and Microsoft Message Queue Services (MSMQ) in Windows NT Server, COM provides better tools for developing enterprise and powerful distributed applications [6].

COM is a platform-independent, distributed, object-oriented system for creating binary software components that can interact and be reused. COM is the foundation technology for Microsoft's OLE (compound documents) and ActiveX (Internet-enabled components) technologies as well as for other component services.

Simply speaking, COM is only a set of standards for building client server applications. Its essence is agreed binary interfaces based on Remote Procedure Calls (RPC) with some wrappers that form the objects and interfaces between the objects [3–6]. The server is a component that provides services or functionality, and the client is the component that calls the services or functionality that are exposed by the server. In this standard, all the functions provided by the server are exposed through interfaces instead of ordinary functions. Fig. 1 shows a client-server application architecture.

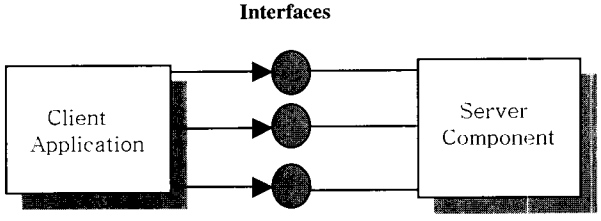



**Fig. 1** A client server application

In the figure, the server component provides three interfaces A, B, and C. Each interface will include a set of functions that can be called by the client application module. The functions provided by the server must be queried and accessed through the interfaces.

This kind of interface engine is simple yet elegant. It brings about tremendous benefits for software developments.

### 2.2 Advantages

The major benefits resulting from COM techniques are software reuse, expandability, upgradability, language independence, integration of thirty party products, database management, and easier building of web based applications. Each of them is illustrated as follows [3–6].

***Software reuse*** Reuse of existing software is very desirable because, in practice, many projects, especially large ones, are continuations of previous projects. Appropriate reuse of existing modules that have been carefully tested and verified can significantly reduce development costs and time and increase productivity. COM certainly facilitates reuse. In COM programming, every functional module is encapsulated into a component with a set of carefully designed interfaces. All the functions provided by the component are clearly specified through the interface, which serves as a "contract." The client software that uses the component is unconcerned with the details of the implementation of the component; instead, it needs only to comply with the requirements specified by the interfaces. In this way, legacy software can be easily reused.

***Software expandability*** When time elapses, most software needs to be expanded to cater for changing requirements and new purposes. When new modules are added, it is highly desired that the existing modules be un-affected. Otherwise, the entire application may collapse. COM provides a framework for meeting this requirement. When appropriately designed, each module of the entire software is very much decoupled, resulting in module independence. Module independence not only facilitates expandability, but also facilitates upgradability as described below.

***Software upgradability*** In contrast to expandability, upgradability here refers to upgrading certain existing modules instead of adding new ones. Traditionally, the upgrading process may cause the version problem, i.e., a problem resulting from utilizing different versions of certain modules. For example, suppose an application is using a module X with version 1.0. Now a newer version of module X with version 2.0 is developed and is to be integrated into the application. Without the COM technique, we would

have to re-compile the entire system and redeploy the entire system. With COM, we can simply redeploy only the upgraded module into the system without recompiling the whole system.

***Language independence*** In many companies, like power software companies, the employees may have different preferences and skills in different languages such as VB, VC, Java, and so on More importantly, different languages have their own strengths and weaknesses. VC is strong for computing, VB is strong for Graphical User Interface (GUI) design, and Java is good for Internet web programming. Most likely, a big project is divided into certain pieces that are implemented by different persons using different languages. The legacy software may be written in any language, too. It is desirable that the different modules built in different languages be smoothly integrated into one enterprise application. COM is perfect for this purpose. Because COM is a binary component, it provides language independence.

***Integration of third party products*** Software engineers dream of building an application just as the circuit designers assemble a circuit. COM provides a standard for doing this. Because COM forces the programmers to comply with the same standards, the components developed by the third party can be conveniently integrated into a company's own product. COM programming not only enables the use of third-party products, but also enables one's product to be easily used by other parties as well. This may increase the market value of a company's products.

***Database management*** Industrial projects commonly involve extensive database management. Database client technologies for the Windows platform have gained a lot of developments in the last decade. Available database application development tools include Open Database Connectivity (ODBC), Microsoft Foundation Classes (MFC) ODBC classes, Data Access Objects (DAO), Remote Data Objects (RDO), Object-Linking and Embedding Database (OLE DB), and ActiveX Data Objects (ADO). These database interfaces provide an easy way to communicate with different and disparate database systems such as desktop databases (e.g., Microsoft Access and FoxPro), relational database servers (e.g., Oracle and SQL server), and so on. A good database interface provides a uniform interface to different database systems, and makes a company's client software independent of the database used. Of all these database technologies, OLE DB and ADO are the two most promising ones and are built using COM techniques. ADO is built on top of OLE DB and is an OLE DB consumer. Applications that use ADO use the OLE DB interfaces indirectly [5].

A COM interface is more robust and flexible than a traditional call-level interface, such as the ODBC interface. Better performance and flexibility can be achieved by using ADO. Another major benefit is that ADO can deal not only with relational databases, but also with non-relational databases like e-mail stores, object databases, and network directories. Essentially, ADO can access any data source for which there is an OLE DB provider. In this way, the client software is more reusable and maintainable.

***Web based applications*** Microsoft Active Server Pages (ASP) and COM greatly facilitate web based applications. Various application modules can be developed into COM components and accessed through VBScript or JavaScript in ASP. In addition, ADO together with scripting technology can greatly facilitate database development on the web.

Additional benefits can be obtained using the latest developments on COM. This includes Distributed Component Object Model (DCOM) and Microsoft Transaction Server (MTS). DCOM is for developing distributed applications, during which you can launch the server that is located on a different machine from the client software. MTS facilitates the building of multitiered applications by providing standard services. MTS handles object instantiation at the server, process and thread management, synchronization of shared resources, and security. MTS helps to build scalable, transaction-oriented, server-side components and applications [4–6]. COM+ is the next evolution of COM and MTS. COM+ offers additional services such as easier handling of the resource management and security. COM+ can be used to develop enterprise-wide, mission-critical, distributed applications based on the Windows 2000 operating system [6]. A good understanding of COM is a good start for COM+.

## 2.3 Available Tools for Developing COM Components

Building COM components using Visual Basic and Java is a lot easier than using Visual C++ because the complexities involved are well handled by Visual Basic or Java itself. When developing COM using C++, Active Template Library (ATL) and MFC classes are two major tools provided by Microsoft. Both provide a set of wrapper classes that encapsulate the routine and common parts needed for COM based programming. ATL is tailored to develop light-weight components, while MFC is better for developing components involving a lot of GUIs. The size of the developed component based on MFC is normally far larger than that based on ATL due to the large size of the MFC DLL.

When developing a server, the housing component can be in either EXE or DLL form. A remote server, which can be launched from a remote machine, needs to be imple-

mented in an EXE form.

Other developing tools like Borland C++ builder can also be used, which assimilates a lot of advantages from Delphi.

### 2.4 Deployment of COM Components

COM has been implemented on various operating platforms such as Apple, UNIX, and Windows. Therefore, we can write client and server applications based on COM technology and deploy them on various systems [4–5].

When deploying the developed components in either DLL or EXE form, we need to appropriately register the components in the registry of the computer where the server is located. This can be done automatically through the installation software or manually by running specific executables like "servername -Regserver" for an EXE server or "regsvr32 servername" for a DLL server.

After the server is successfully registered, it can be examined through the Microsoft tool OLE/COM object viewer. We can instantiate a server component through this viewer and treat this as the first step for debugging a client-server application when needed.

## 3. Applications of COM for Power System Software Development

We have seen so far that significant benefits can be gained when COM is appropriately used for a wide range of software developments. This section presents possible applications of COM to power system software developments.

### 3.1 Sparse Matrix (SP) Application

Sparse matrix manipulation including memory allocation, LR factoring, and so on plays a very important role in power system analysis and simulation software developments. It is extensively used in the forming of admittance matrices of a power system network and solving of the power flow equations. Various kinds of sparse matrices have been developed by different authors. A software package written in C as described in [7] is an example. It is rather inconvenient for us to use such a package because no standards exist for describing the functions provided by the package, let alone the numerous macros decorated by #defines.

It will be indeed desirable if we develop a COM component to wrap the existing functionality and describe all the functions in the interface file, as shown in Fig. 2.

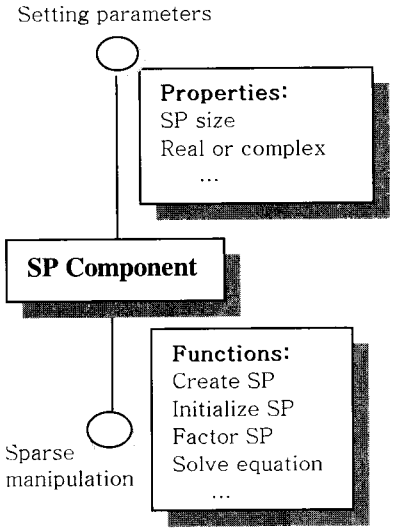



**Fig. 2** SP component

In the figure, the Setting parameters set pertinent SP parameters. Real or complex indicates whether the element is a real or complex number. Interface Sparse manipulation implements desired SP functions.

Next we propose a novel yet efficient approach for encapsulating the existing sparse matrix package into a COM component. The following steps are very general and can be used for encapsulating other types of existing software too.

Step 1: The existing software is analyzed into functional modules. The inputs and outputs of the modules are identified. Each module is then represented by a C++ class. Generally speaking, a module can be represented as a black box as shown in Fig. 3.

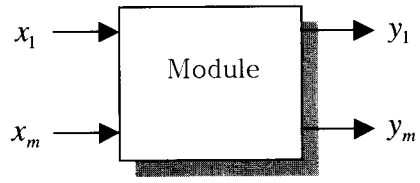



**Fig. 3** Block representation of a functional module

In the figure, $x_1$, ..., and $x_m$ represent the first to $m^{th}$ inputs to the module, and $y_1$, ..., and $y_n$ represent the first to $n^{th}$ output of the module. The output can be concrete variables, operation or both such as creating a sparse matrix, returning a non-zero element, factoring the sparse matrix, solving a linear equation, and so on. This module can be converted into a C++ class as follows.

```
class ClassModule
 {
 private:
        TypeX1  x1;
        ...
        TypeXm  xm;
 public:
        HRESULT PutX1(TypeX1  x1);
        ...
        HRESULT PutXm(TypeXm  xm);
        HRESULT GetY1(TypeY1*  y1);
        ...
        HRESULT GetYn(TypeYn*  yn);
 };
```

where, TypeX1, TypeXm, TypeY1, TypeYn, and so on are data type specifiers. PutX1(), PutXm(), and so on assign values to private members $x_1$, $x_m$, and so forth. GetY1() returns the value of $y_1$, GetYn() obtains the value of $y_n$, etc. They may also perform certain defined operations. HRESULT is the return data type of the public methods.

Step 2: The C++ class is encapsulated into COM components. A COM component implements a set of interfaces that support specific methods. The above class can be converted into a component whose interfaces are shown as follows.

```
Interface IClassModule: IUnknown
{
[helpstring("method PutX1")] HRESULT PutX1([in] TypeX1
x1);
...
[helpstring("method PutXm")] HRESULT PutXm([in]
TypeX1 xm);
[helpstring("method GetY1")] HRESULT GetY1([out]
TypeY1* y1);
...
[helpstring("method GetYn")] HRESULT GetYn([out]
TypeYn* yn);
};
```

A C++ class may be represented by more than one interface if needed, each of which can be in a form similar to IClassModule as shown above.

When implemented, one or more interfaces can be supported by each COM class. The following steps show how to build a COM component implementing the above interface IClassModule. The COM class implementing the interface is named as CClassModule.

1). Build a C++ project: Here, Visual C++ 6.0 ATL COM class wizard can create most of the needed codes including codes for the interface definition file (i.e., .idl file), registry files, etc. [6]. The project needs to include the definition and implementation files of class ClassModule. The project will also include a standard COM class CClassModule.

2). Add all the supported methods as follows.

```
class ATL_NO_VTABLE CClassModule:
{
...
public:
        STDMETHOD( PutX1)(TypeX1  x1) ;
        ...
       STDMETHOD( GetYn)(TypeYn* yn);
}
```

3). Add a private member of type CClassModule to the created CClassModule class.

```
class ATL_NO_VTABLE CClassModule:
{
private:
        ...
        ClassModule m_ClassModule;
}
```

4). Add implementation of the supported methods. As examples, implementation of PutXm(TypeXm $x_m$) and GetYn(TypeYn* $y_n$) is shown as follows.

```
STDMETHODIMP CClassModule::PutXm(TypeXm  xm)
{
        HRESULT hr = m_ClassModule.PutXm( xm);
        //put additional error handling codes here, if wanted.
        return hr;
}
STDMETHODIMP CClassModule::GetYn(TypeYn*  yn)
{
        HRESULT hr = m_ClassModule.GetYn( yn);
        //put additional error handling codes here, if wanted.
        return hr;
}
```

Then the source files can be compiled to generate the components.

### 3.2 Developing a Genetic Algorithm Component Using COM for Power System Planning Applications

GA is a simple yet powerful tool for finding the global solution to an optimization problem. It has found wide application in various power system areas such as unit commitment, system expansion, capacitor placement, feeder automation, and so on. GA includes the following steps: fitness evaluation, parent selection, crossover, and mutation.

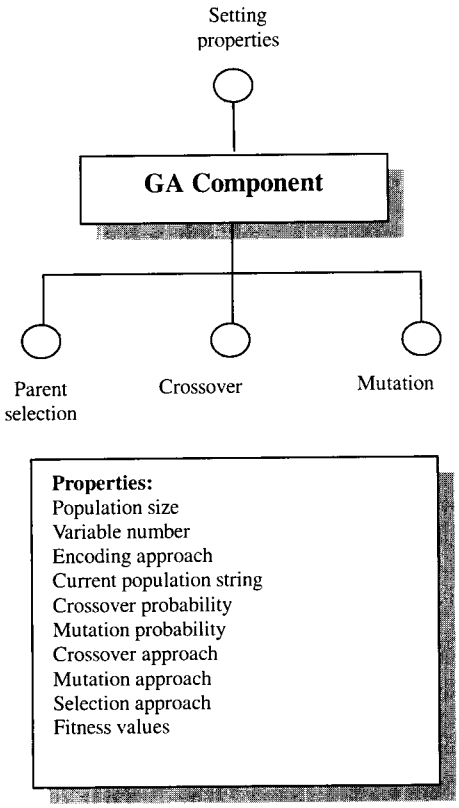



**Fig. 4** A GA component

All the above steps except fitness evaluation are common among different applications [1]. To improve productivity, it is desired that these common steps be developed once and reused by various applications. Developing a COM component is a promising way to accomplish this. A component structure is shown in Fig. 4.

In the figure, the interface setting properties set the values of a set of properties for input and output functions. Population size is the population in each generation, variable number is the number of variables used in the GA, encoding approach can be integer coding or binary coding, current population string is the latest strings in the current generation, crossover approach can be independent (each variable has its own crossover independent of each other) or dependent (the entire population string consisting of all the variables has only one crossover), mutation can be independent or dependent, selection approach can be roulette wheel or ranking based, and fitness values need to be obtained outside the GA component depending on the actual problem. For example, for system expansion, the fitness function may be defined as the minus of the total expansion cost. For unit commitment, it can be defined as the minus of the total cost of the operation of the committed units.

The interface parent selection implements the selection process. The interface crossover implements the crossover process. The interface mutation implements the mutation process. All these processes correspond to the according property values. Different types of power system applications may need different property values. Once the component is developed, it can be easily integrated into power system application software.

### 3.3 Developing a Fault Location Component Using COM

Tremendous efforts have been made in the past for developing various types of fault location algorithms (FLA). An efficient and practical way to implement these algorithms is needed. Fig. 5 shows the proposed component for fault location (FL) applications.

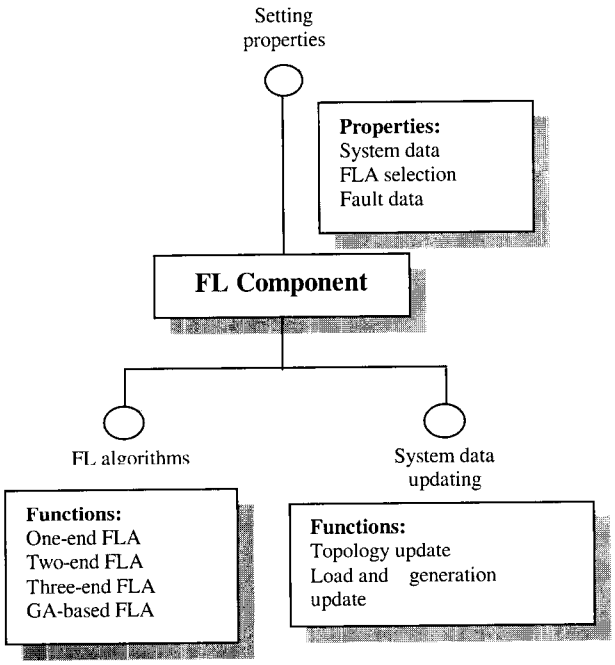



**Fig. 5** The fault location component

In the figure, interface FL algorithms implement various types of fault location algorithms [1]. System data uptating includes functions of updating topology, load, and data generation Setting properties pass the system data, FLA selection, and the fault data(e.g., digital fault recorder data and relay data) to the component.

### 3.4 Developing Universal Relay Interfaces (URI) for IEDs

In the USA, diverse types of Intelligent Electronic Devices (IED) like digital relays and digital fault recorders have been installed at power substations for monitoring and recording specified analog data like voltage and current signals and digital data like circuit breaker status.

Normally, IED setting, data uploading, and control are managed by the manufacturer's proprietary software. Application software controlling the IED is quite difficult to develop since the protocol and data saving format must be exactly known. A COM based embedded system may be schemed to interpret the details specific to the protocol. Any new application software can be protocol-free and easily developed using this embedded system as a common interface. This can reduce the costs for developing customized applications. The enhanced capability will put IED products in a more competitive market position. Fig. 6 shows such a design.

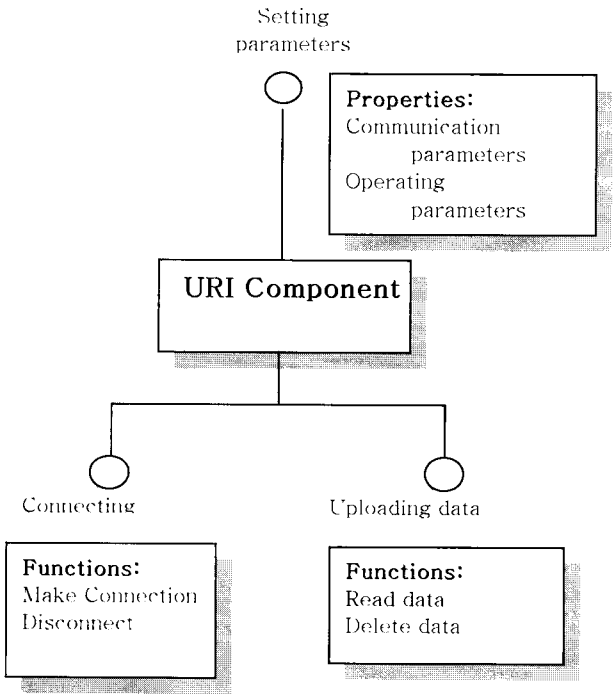



**Fig. 6** The URI component

In the figure, interface connecting makes the communication connection between the client application software and the IED or disconnects the connection. Interface setting parameters set the IED parameters including communication parameters such as parity, user i.d., password, baud rate, byte size, stop bit, communication port, and protocol (e.g., Integrated Communication Protocol: INCOM, and ASCII) and operating parameters such as tripping criterion (e.g., 1.2 p.u. for first zone distance protection), operation mode (e.g., over-current and distance protection), and recording options (e.g., which signals to record). Interface uploading data reads or deletes the recorded data from the IED. Equipped with such a universal interface, communication and control of the IED by the application software, which is usually located at a central control office, can be very much facilitated.

Other COM applications may include developing modules like power flow and power quality analysis as well as integrating the third party power system one-line geographical diagram tool into the application software.

## 4. Conclusions

COM is a well established client server standard for building modular and expandable components. Possible applications in power system software development such as sparse matrix manipulation, building of a GA component for power system applications, and building of a fault location component have been presented. Significant benefits can be gained by using COM. Additional research and practical experience will be useful in judging the appropriateness of and further aiding the use of COM in power software community.

## Acknowledgements

This paper was partially supported by Wonkwang University.

## References

[1] Mladen Kezunovic and Yuan Liao, "Development of a Fault Locating System using Object-Oriented Programming," IEEE Power Engineering Society Winter Meeting, Vol. 2, 2001, pp.763-768.

[2] R.C. Dugan and T.E. McDermott, "Design of interfaces for Power Systems Analysis Components," IEEE Power Engineering Society Summer Meeting, Vol. 2, 1999, pp.1164-1169.

[3] Dale Rogerson, Inside COM, Microsoft Press, February 1997.

[4] Jonathan Bates, *Creating Lightweight Components with ATL*, Sams Publishing, 1999.

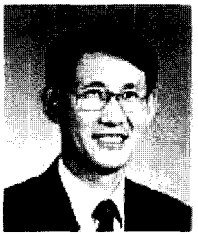
[5] Lyn Robison, *Teach yourself Database Programming with Visual C++ 6*, Sams Publishing, 1999.

[6] "Microsoft Developer Network (MSDN) Library," Microsoft Corporation, July 2000.

[7] Kenneth S. Kundert and Alberto Sangiovanni-Vincentelli, *Sparse User's Guide, A Sparse Linear Equation Solver*, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, July 1, 1998.

**Yuan Liao**

He received his Ph.D. degree from Texas A&M University, College Station, Texas, in Electrical Engineering in 2000. He joined ABB as a consulting R&D engineer in 2000. His research interests include power system monitoring, protection, control, and system planning. He can be reached at ABB Corporate Research Center, Raleigh, NC, 27606, USA.



**Jong-Beom Lee**

He received B.S., M.S., and Ph.D. degrees in Electrical Engineering from Hanyang University, Korea, in 1981, 1983, and 1986, respectively. He worked at the Korea Electrotechnology Research Institute from 1987 to 1990. He was a visiting scholar at the Technical University of Berlin, Germany, in 1993, the City University, UK, in 1995, and Texas A&M University in 1997. He is currently a professor in the Department of Electrical, Electronic, & Information Engineering, Wonkwang University, Korea. His current research interests are power system operation, transient analysis of power cable system, protective relaying, application of neural networks, and application of fuzzy logic to power systems. He is a member of KIEE and IEEE.