효율적인 OPC Client 생성을 위한 ActiveX 기반 컴포넌트의 설계 및 구현

Design and Implementation of an ActiveX based Component For Efficient Generation of OPC Client

김 종 환*, 심 민 석, 이 명 재
(Jong-Hwan Kim and Min-Suk Sim, and Myeong-Jae Yi)

Abstract : OPC has made it to improve the development of control and monitoring software. But it is difficult to understand COM/DCOM model that is a base technology of OPC and complex communication setting between OPC Server and OPC Client. Therefore, in this paper, we design and implement ActiveX based components that are enable to solve these problems. Implemented components provide a method for simple communication between OPC Server and OPC Client and a GUI environment for easy and fast setting of communication information. Also, they provide the architecture for efficient management of OPC Server's data. By using implemented component, easy development and efficient maintenance of OPC Client can be supported.

Keywords : OPC, COM, DCOM, ActiveX

I. 서론

산업이 발전함에 따라 생산 현장의 장비들은 필드버스, 이더넷과 같은 산업용 네트워크를 통해 크고 복잡한 설비를 이루게 되었고, 설비를 이루고 있는 장비들의 제어를 위해 PC 기반의 시스템이 도입되었다. 그러나 이런 시스템들은 벤더의 드라이버에 종속적이기 때문에 시스템의 개발 및 유지 보수가 어렵고, 매우 고가인 문제점을 가지고 있었다.

이를 해결하기 위해서 프로세스 컨트롤에 대한 논의가 시작되었고, 그 결과 OPC(OLE for Process Control)[1]가 발표되었다. OPC는 마이크로소프트의 COM/DCOM[2]을 기반으로 PC 기반의 소프트웨어 컴포넌트들이 데이터를 교환하는데 필요한 표준 인터페이스를 정의한다. OPC의 등장으로 인해 더 이상 벤더에 종속적인 드라이버를 사용하지 않아도 장비를 제어할 수 있게 되었고, 그에 따른 소프트웨어의 유지보수 비용이 줄어들게 되었다. 이런 장점들로 인해 OPC는 프로세스 컨트롤 분야에서 표준 인터페이스로 자리를 잡아가게 되었다.

OPC로 인해 제어 소프트웨어 개발에 있어 많은 부분이 개선이 되었지만 아직 OPC client 어플리케이션과 OPC server 사이의 통신에는 많은 어려움이 존재하고 있다. 우선 OPC 시스템을 구축하기 위해서는 OPC의 기본 기술이 되는 COM/DCOM에 대한 전문적인 지식을 가지고 있어야 하고, 그에 대한 능숙한 프로그래밍 능력도 필요로 한다. 또한 COM/DCOM을 사용하기 위해서는 C/C++와 같은 언어를 사용해야 하므로 개발자는 이런 프로그램 언어에 대한 전문적인 지식도 함께 가지고 있어야 한다. 그리고 OPC client와 OPC server 사이의 상호 작용과 OPC 시스템의 기본 구조에 대한 이해도 필요로 한다.

이에 본 연구에서는 이런 문제점들을 해결하기 위해서 마이크로소프트사의 COM/DCOM 기반의 OPC 표준을 지원하는 ActiveX[3] 기반의 컴포넌트를 설계하고 구현한다. ActiveX 기반의 컴포넌트를 통하여 OPC client 개발자는 COM/DCOM에 대한 전문적인 지식이 없이도 쉽게 OPC client를 개발할 수 있고, OPC 시스템의 구조에 대한 이해가 부족하더라도 쉽게 OPC server에 접근하고 조작하는 것이 가능하다. 이를 통해 OPC client의 빠른 개발과 효율적인 유지보수를 지원한다.

본 논문의 구성은 다음과 같다.

제 2장에서는 본 연구를 진행함에 있어 필요한 기술적 배경과 관련 연구에 대해서 알아보고, 제 3장에서는 본 연구에서 구현하는 OPC-DA client 컴포넌트에 대해서 알아본다. 제 4장에서는 개발된 컴포넌트를 이용한 OPC client의 구현에 대해서 알아보고, 마지막으로 제 5장에서는 결론 및 향후 연구 과제에 대해서 논의한다.

II. 관련 연구

이 장에서는 본 연구를 수행함에 있어 필요한 기술적 배경과 기존의 관련 연구에 대해서 기술한다.

1. OPC(OLE for Process Control)

OPC는 프로세스 제어 분야에서 마이크로소프트사의 COM/DCOM 기술을 사용하여 공장 자동화 어플리케이션들이 다양한 장비들(DCS, PLC, 등)과 통신을 하는데 필요한 객체, 인터페이스, 메소드들을 정의하는 개방형 표준 인터페이스를 정의하는 것으로, WinSEM(Windows for Science, Engineering, and Manufacturing) 그룹에서 프로세스 컨트롤에 대한 표준을 개발하기 위한 노력의 일환으로 시작되었고, 1996년 처음 발표된 이후 사용자 어플리케이션과 장비들 사이의 인터페이스를 정의하는 표준으로 자리 잡고 있다. OPC 표준은 OPC client(사용자 어플리케이션)과 OPC server(DCS, PLC와 연결된 I/O server) 사이의 인터페이스를 정의하는 것으로, 높은 수준의 상호 운용성을 제공한다. 현재 가장 기본

* 책임저자(Corresponding Author)
논문접수 : 2005. 2. 24., 채택확정 : 2005. 9. 12.
김종환, 심민석, 이명재 : 울산대학교 컴퓨터정보통신공학과
(bearknight@mail.ulsan.ac.kr/sms@mail.ulsan.ac.kr/ymj@ulsan.ac.kr)
※ 본 연구는 산업자원부 · 울산광역시 지원 울산대학교 네트워크 기반 자동화 연구센터의 지원에 의한 것임.

이 되는 표준인 data access 표준을 비롯한 historical data access, XML data access, alarms & events 등의 모두 10가지의 표준[1]이 정의되어 있다.

2. ActiveX

윈도우 개발 환경에서는 개발자가 스스로 컨트롤을 제작해서 사용할 수 있었으며, 이 컨트롤을 커스텀 컨트롤이라고 불렀고, 컴포넌트 개발 환경이 COM 기반으로 바뀌면서부터 OLE컨트롤이라고 불리어지기 시작했다. 이후 점차 확산되는 네트워크 환경에 적합하도록 개선하는 작업이 시작되었고, 그 결과 ActiveX[3]가 개발되었다. ActiveX는 컨트롤이 필수 인터페이스만 지원함으로써 컨트롤을 경량화하여 네트워크 환경에서 효율적으로 사용할 수 있도록 하였다. 꼭 네트워크 환경에서만 사용할 수 있는 것이 아니라, OLE 컨트롤을 사용할 수 있는 컨테이너를 지원하는 환경에서라면 ActiveX를 사용할 수 있어 기존의 OLE 컨트롤보다 활용성이 더 좋아지게 되었다.

OPC client 개발을 지원하는 도구들은 ActiveX와 .NET 컴포넌트의 형태를 가지게 되는데, .NET 컴포넌트를 사용하기 위해서는 반드시 .NET 프레임워크가 설치되어 있어야 하며 그렇지 않은 환경에서는 사용할 수가 없다. 이에 반해 ActiveX 컨트롤은 visual studio 6.0 이하의 개발 환경뿐만 아니라 .NET 개발 환경 하에서도 특별한 설정 없이 사용될 수 있다는 장점이 있다. 따라서 본 연구에서 구현하는 컴포넌트는 ActiveX를 기반으로 구현되었다.

3. OPC 관련 연구

OPC는 1996년 발표된 이후로 공장 자동화 분야에서 필드버스와 함께 가장 많은 관심을 받고 있는 분야 중 하나이며, 다양한 연구가 이루어지고 있고, 상당수의 관련 제품들이 출시되고 있다. OPC에 대한 제품군은 OPC server, OPC client, developer tools & services의 3가지 형식으로 분류되며, 본 연구에서 구현하는 컴포넌트는 developer tools & services의 분류에 속하게 된다. 국내에는 소개 된지 오래되지 않아서 OPC 관련 연구들은 이제 시작 단계에 있고, 관련 제품들도 대부분 수입한 제품들이거나 개발 중에 있다.

초기에는 OPC 명세를 활용하는 연구와 기존 시스템에 OPC를 적용하는 연구들이 주를 이루었다. 대표적으로 OPC 표준을 기반으로 하고 XML 기반의 디바이스 기술 문서를 사용하여 디바이스 드라이버들 간의 통신 인터페이스에 대한 표준 명세라고 할 수 있는 driver server의 개발에 대한 연구[4]와 기존의 SCADA(Supervisory Control and Data Acquisition) 시스템에 OPC를 적용하여 시스템의 유지 보수성을 향상시키고, 윈도우의 "plug and play" 개념을 도입하여 기존 시스템의 지속적인 개발과 업그레이드를 지원하도록 하는 방법에 대한 연구[5]가 있다.

이후에는 다양한 분야에 OPC를 적용 또는 연동하는 방안에 대한 연구가 이루어졌다. 대표적으로 플랜트 내부에 대해서만 서비스를 제공하는 OPC 시스템의 웹 서비스 지원을 위한 게이트웨이의 개발에 대한 연구[6]와 웹 서비스를 이용하여 OPC 시스템이 SMS(Short Message Server)를 지원하도록 하는 방안에 대한 연구[7]가 있다.

OPC client 개발을 지원하는 ActiveX 기반의 컴포넌트는 이미 기존에 다수가 존재하고 있으나 제각기 그 한계를 가지고 있다. Merz사의 "easy OPC-DA"[8]는 OPC server와의 연결 및 아이템의 추가와 아이템의 데이터 접근과 같은 기본적인 기능들을 제공하고 있으나, OPC server와의 통신설정을 자세히 할 수 없고, 아이템의 관리에 제약을 가지는 한계를 가지고 있다. 이에 비해 Technosoftware사의 "TsOPC client framework"[9]는 비교적 다양한 기능을 제공하고 있으나, GUI 환경을 제공하지 않아 프로그래밍에 익숙하지 않은 사용자가 사용하기에 불편한 점이 있다. 이는 Merz사의 "easy OPC-DA"도 가지고 있는 한계이다. Eldridge Engineering사의 "OPC controls"[10]는 앞선 두 컴포넌트와 달리 GUI 환경에서 OPC server를 쉽게 검색하고, 사용자가 편리하게 아이템을 관리할 수 있도록 지원하고 있지만, OPC server와의 연결에 필요한 그룹에 대한 설정을 할 수 없다는 단점을 가지고 있다. 이에 반해 Softing의 "OPC DA client control"[11]은 OPC group에 대한 설정을 지원하지만, OPC-DA 2.0 명세에서 제공하는 private 그룹을 public 그룹으로 변경하는 작업을 지원하지 않는 등 public 그룹에 대한 자세한 설정을 지원하지 않는다. 이에 본 연구에서 구현하는 컴포넌트는 앞선 컴포넌트들의 부족한 점을 보완하여, OPC server와의 통신에 필요한 다양한 기능과 속성을 제공하고, GUI 환경의 제공을 통해 사용자 편의성을 지원하도록 구현되었다.

III. OPC-DA Client 컴포넌트

이 장에서는 본 연구에서 구현하는 OPC-DA client 컴포넌트에 대해서 알아본다. 본 연구에서 구현하는 DA client 컴포넌트는 그림 1과 같은 환경에서 실행이 된다. 구현하는 OPC-DA client 컴포넌트는 직접 필드 디바이스에 접근해서 데이터를 가져오거나 쓰는 것이 아니라 각 필드 디바이스에 연결되어 있는 OPC server들을 통해서 필드 디바이스의 데이터에 접근하게 된다. OPC-DA client 컴포넌트는 OPC-DA 표준 인터페이스에 따라 OPC server와 통신을 하며, visual basic 또는 visual C++ 프로그래머가 직접 OPC server에 접근하지 않고도 쉽고 간단하게 OPC client를 작성할 수 있도록 한다.

1. 컴포넌트의 구조

본 연구에서 구현하는 컴포넌트는 microsoft visual studio 2003 환경에서 visual C++를 이용하여 구현되었다. 그림 2는 구현된 컴포넌트의 클래스와 인터페이스들 사이의 관계를 보여준다.

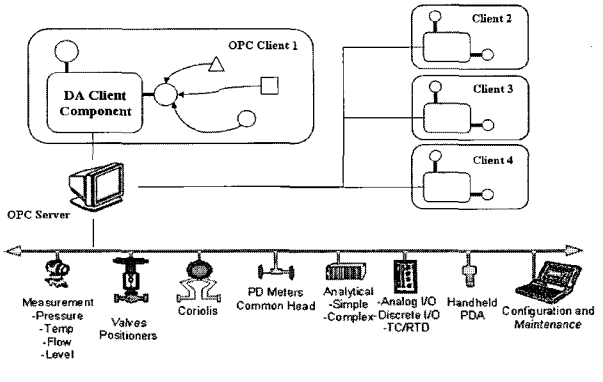


그림 1. 컴포넌트의 실행환경.

Fig. 1. Component execution environments.

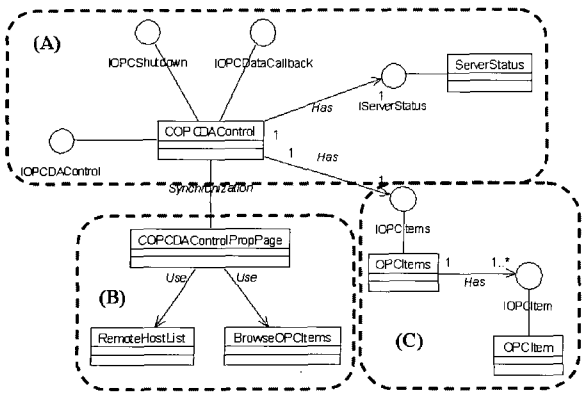


그림 2. 컴포넌트의 구조.
Fig. 2. Component architecture.

표 1. 컴포넌트의 속성.
Table 1. Properties of component.

속성	속성에 대한 설명
RemoteHost	OPC server가 존재하는 호스트.
OPCDASpec	연결할 OPC server의 명세 버전.
OPCServerName	연결할 OPC server의 이름
ServerStatus	OPC server의 상태를 나타내는 속성. 그림 2의 server status 객체를 반환한다.
Timeout	원격 호스트와의 연결을 감시하는 주기.
Monitor	속성 값이 1이면 OPC server와의 연결을 감시.
MonitorTimeout	OPC server와의 연결을 감시하는 주기.
IsConnected	속성 값이 1이면 OPC server에 연결되어 있음을 표시.
GroupName	사용할 OPC group의 이름
Active	속성 값이 TRUE로 설정되면 그룹 내의 active 아이템의 데이터가 변하게 되었을 때, 클라이언트가 변한 데이터를 수신한다.
UpdateRate	비동기 방식으로 데이터를 수집할 때 데이터가 갱신되는 주기.
Deadband	OPC server의 아이템 값이 변하게 되었을 때 클라이언트가 변한 값을 수신할 확률을 나타내며, 0부터 100사이의 값만 가진다. 속성값이 0이면, 클라이언트는 항상 변경된 값을 수신한다.
IsPrivate	속성 값이 0이면 현재 그룹이 private 그룹임을 나타냄.
UseAsync	속성 값이 1이면 OPC server로부터의 콜백 이벤트를 처리한다.
Items	OPC item들의 집합을 나타내는 속성. 그림 2의 OPC items 객체를 반환한다.

그림 2의 (A)는 OPC server와의 통신을 수행하는 모듈로써 OPC server와의 연결 및 연결 해제, 그룹과 아이템의 생성, OPC server 내의 아이템의 데이터 관리 등의 작업을 수행한다. 그림 2의 (B)는 OPC server와의 통신에 필요한 정보들을 설정할 수 있도록 GUI 환경을 제공하는 GUI 모듈을 나타낸다. 그림 2의 (C)는 OPC server내에 생성된 아이템과 각 아이템의 데이터를 관리하는 아이템 관리 모듈을 나타낸다. 이후에는 각 부분에 대해서 차례대로 알아보도록 한다.

2. OPC Server와의 통신을 수행하는 모듈

이 절에서는 그림 2의 (A) 부분에서 제공하고 있는 기능들에 대해서 자세히 설명하도록 한다. 그림 2의 (A)는 구현하는 컴포넌트의 가장 기본이 되는 부분이며, 실질적인 컴포넌트를 나타내는 부분이기도 하다.

2.1 컴포넌트의 속성

본 연구에서 구현하는 컴포넌트는 OPC server와의 연결에 필요한 정보들을 속성의 형태로 제공하여 사용자가 관련 정보를 쉽게 사용할 수 있도록 하고 있다. 컴포넌트에서 제공하는 속성은 연결할 OPC server에 대한 정보를 나타내는 속성과, OPC server 내에 생성할 그룹에 대한 정보를 나타내는 속성, 그리고 그룹 내에 생성할 아이템에 대한 정보를 나타내는 속성으로 나눌 수 있다.

2.2 컴포넌트에서 제공하는 메소드

본 연구에서 구현하는 컴포넌트는 OPC server와의 연결 및 데이터 관리에 있어 편의성을 제공하기 위하여 표 2에서 보는 것과 같은 메소드들을 제공하고 있다. 이중 connect 메소드는 OPC server와의 연결에 필요한 복잡한 과정들을 모두 수행함으로써 사용자에게 편의성을 제공하고 있다. 그림 3은 connect 메소드 내부에서 수행되는 작업들을 보여주고 있다.

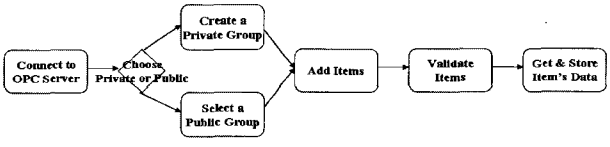


그림 3. Connect 메소드에서 수행하는 작업.
Fig. 3. Sequence of jobs in connect method.

표 2. 컴포넌트의 메소드.
Table 2. Method of component.

메소드	메소드에 대한 설명
Connect	그림 3에 보여지는 동작을 수행.
Disconnect	OPC server와의 연결을 해제하고 관련 리소스들을 해제하는 동작을 수행.
Change Private To Public	Private 그룹을 public 그룹으로 변경하는 메소드.
Refresh	현재의 OPC server의 아이템의 데이터를 가져오는 메소드. 파라미터 값이 0일 때는 동기적으로 데이터를 수신, 1일 때는 비동기적으로 데이터를 수신한다.
Update	Items 속성에 설정되어 있는 값을 OPC server의 아이템에 적용시키는 메소드. 파라미터 값이 0일 때는 동기적으로 아이템의 데이터를 수정, 1일 때는 비동기적으로 아이템의 데이터를 수정한다.

표 3. 콜백 인터페이스의 메소드.
Table 3. Methods of callback interfaces.

메소드	메소드에 대한 설명
IOPCDataCallback 인터페이스	
OnDataChange	아이템의 데이터가 변경되었을 때 호출되며, 변경된 데이터를 컴포넌트의 각 item객체에 저장하도록 구현된다.
OnReadComplete	비동기적으로 아이템의 데이터를 읽어왔을 때 호출되며, 컴포넌트의 해당 item 객체에 데이터를 저장한다.
OnWriteComplete	비동기적인 아이템의 데이터 수정 작업이 완료되었을 때 호출되며, 작업의 오류여부를 판별하도록 구현된다.
OnCancelComplete	비동기적으로 수행되는 데이터를 읽어오는 작업 또는 데이터 수정 작업이 취소되었을 때 호출된다.
IOPCShutdown 인터페이스	
ShutdownRequest	OPC server로부터 연결이 종료되기 전에 호출되며, 관련 리소스들을 해제하도록 구현된다.

```
void CJOPCDAControl::DoPropExchange(CPropExchange*
pPx){
    ExchangeVersion(pPX,
            MAKELONG(_wVerMinor, _wVerMajor));;
    COleControl::DoPropExchange(pPX);
    PX_String(pPX, _T("OPCServerName"),
            m_OPCServerName);
    ....
}
```

그림 4. 속성 지속성의 처리.
Fig. 4. Property persistence handling.

그룹의 생성에서 기존에 존재하고 있는 public 그룹을 선택하였을 때, 기존 그룹의 설정을 사용할 수도 있지만 현재 클라이언트가 원하는 설정으로 변경할 수도 있는 기능을 제공한다.

2.3 콜백(callback) 인터페이스의 구현

컴포넌트는 기본적으로 동기적으로 아이템의 데이터를 읽고 쓰는 작업을 수행한다. 그러나 사용자가 원할 시에 컴포넌트는 비동기적으로 OPC server에서 보내오는 콜백 이벤트를 받을 수 있어야 하기 때문에 IOPCDataCallback 인터페이스와 IOPCShutdown 인터페이스를 구현해야 한다. IOPCShutdown 인터페이스는 OPC server가 종료하기 전 OPC client에게 종료 사실을 알려주는 역할을 수행하고, IOPCDataCallback 인터페이스는 OPC server의 데이터가 변경되었을 때 OPC client에게 변경된 데이터를 보내주는 역할을 수행한다. 표 3은 각 인터페이스의 메소드와 각 메소드의 기능 및 구현 내용을 보여준다.

2.4 속성 지속성

속성 지속성은 컴포넌트의 속성을 컴포넌트를 조작하는 프로그램보다 오래 유지시키는 것을 의미한다. visual basic과 같이 프로그램의 디자인 타임과 런타임을 가지는 개발 프로그램에서 컴포넌트를 사용할 때 속성 지속성이 구현되어 있지 않다면 디자인 타임에서 설정한 속성 값이 런타임 시에 반영이 되지 않는 문제가 발생하게 된다. 그림 4는 속성 지속성 처리를 하는 간단한 예제를 보여준다.

속성 지속성을 처리하기 위해서 PX_String(), PX_Long(), PX_Short()와 같은 PX_XXX()메소드를 사용한다. 그림 4의 PX_String() 메소드의 두 번째 파라미터인 OPCServerName은 지속성 처리를 원하는 속성의 이름을 나타내고, 세 번째 파라미터인 m_OPCServerName은 속성 값을 가지고 있는 변수를 나타낸다.

3. GUI 모듈

이 절에서는 그림 2의 (B) 부분에 대해서 자세히 설명하도록 한다. 그림 2의 (B)는 구현하는 컴포넌트가 OPC server와의 연결 설정에 필요한 속성들을 쉽게 설정할 수 있도록 제공하는 GUI 환경의 구조를 보여준다.

그림 5는 컴포넌트에서 제공하는 GUI 클래스들을 보여준다. (A)는 표 1에 기술된 속성들을 쉽게 설정할 수 있도록 GUI 환경을 제공하는 부분으로 OPC server를 자동으로 검색해주는 기능도 제공한다. (A)에서 설정된 속성들은 컴포넌트 클래스와 동기화를 시켜주어야 프로그램의 런타임에 적용될 수 있다. 동기화를 위해서 DDP_CBString(), DDP_Text() 메소드와 같은 DDP_XXX() 메소드를 이용한다. 그림 6은 동기화를 시켜주는 예제를 보여준다. DDP_CBString() 메소드는 문자

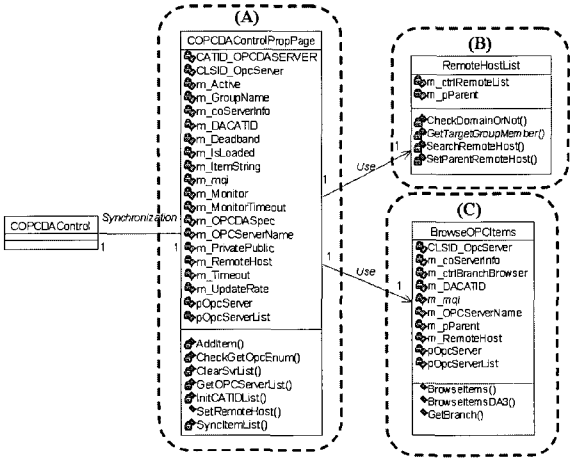


그림 5. GUI 환경.
Fig. 5. GUI environment.

```
void CJOPCDAControlPropPage::DoDataExchange(
                                CDataExchange* pDX) {
    DDP_CBString(pDX,
            IDC_COMBO_OPCSERVERLIST,
            m_OPCServerName,
            _T("OPCServerName"));
    ....
}
```

그림 6. 속성 동기화.
Fig. 6. Property synchronization.

열 속성을 동기화 시켜주는 역할을 수행한다. 메소드의 두 번째 파라미터는 GUI에서의 속성 값을 받아들이는 컨트롤의 ID이고, 세 번째 파라미터는 컨트롤의 데이터를 저장하는 변수를 나타낸다. 마지막 파라미터는 GUI의 컨트롤과 동기화를 시켜줄 컴포넌트의 속성 이름을 나타낸다.

그림 5의 (B)는 연결할 OPC server가 존재하는 원격 호스트를 검색하여 보여주는 부분으로 사용자가 편리하게 원격 호스트를 선택할 수 있도록 도와준다.

그림 5의 (C)는 OPC client에서 사용할 아이템들을 쉽게 선택할 수 있도록 지정한 OPC server의 address space를 검색해서 보여주는 기능을 수행하며, OPC-DA 명세 2.0버전과 3.0버전의 각기 다른 인터페이스를 모두 지원함으로써 버전에 상관없이 OPC server의 address space도 검색하는 것이 가능하다.

4. 아이템 관리 모듈

이 절에서는 그림 2의 (C) 부분에 대해서 설명하도록 한다. 컴포넌트에서는 OPC server에 연결 후에 추가한 아이템과 아이템의 데이터를 효율적으로 관리할 수 있는 기능을 제공한다. client가 사용하는 아이템은 다수가 될 수 있고, 그 아이템의 데이터(bandwidth, value, id, 등)도 OPC-DA 명세에 기술된 만큼 존재하므로 이런 형태의 데이터를 관리하기 위해 본 연구의 컴포넌트에서는 그림 7과 같은 구조로 아이템을 관리한다.

OPC item 객체는 OPC server 내에 존재하는 하나의 아이템의 데이터를 저장하는 객체로 아이템이 가질 수 있는 다양한 형식의 데이터를 저장하는 데이터 저장 객체이다. client가 다수의 아이템들을 사용할 수 있기 때문에 OPC item 객체도 다수가 될 수 있다. 이것을 관리하기 위해 OPC items 객체를 사용한다. OPC items 객체는 객체들을 관리하는 컬렉션 객체로 다수의 OPC item 객체를 추가, 삭제할 수 있는 기능을 제공한다. 각 객체들은 client가 직접 접근할 수 없고 컴포넌트를 통해서만 접근 가능하도록 구현된다.

IV. 컴포넌트를 이용한 OPC client 구현

이 절에서는 본 연구에서 구현한 컴포넌트의 유용성을 보이기 위해, 컴포넌트를 이용하여 OPC client를 제작해보도록 하겠다. OPC client는 microsoft visual studio 6.0 환경에서 visual basic 을 이용하여 제작될 것이며, 다음의 순서를 거쳐 제작된다.

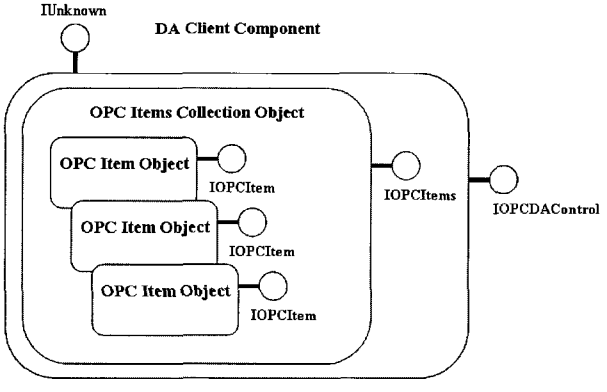
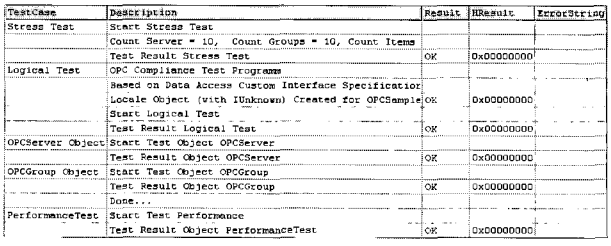


그림 7. 아이템 관리 구조.
Fig. 7. Structure of item management.



TestCase	Description	Result	HResult	ErrorString
Stress Test	Start Stress Test			
	Count Server = 10, Count Groups = 10, Count Items			
	Test Result Stress Test	OK	0x00000000	
Logical Test	OPC Compliance Test Programm			
	Based on Data Access Custom Interface Specificatior			
	Locale Object (with IUnknown) Created for OPCSample	OK	0x00000000	
	Start Logical Test			
	Test Result Logical Test	OK	0x00000000	
OPCServer Object	Start Test Object OPCServer			
	Test Result Object OPCServer	OK	0x00000000	
OPCGroup Object	Start Test Object OPCGroup			
	Test Result Object OPCGroup	OK	0x00000000	
	Done...			
PerformanceTest	Start Test Performance			
	Test Result Object PerformanceTest	OK	0x00000000	

그림 8. OPC server 테스트 결과.
Fig. 8. OPC server test result.

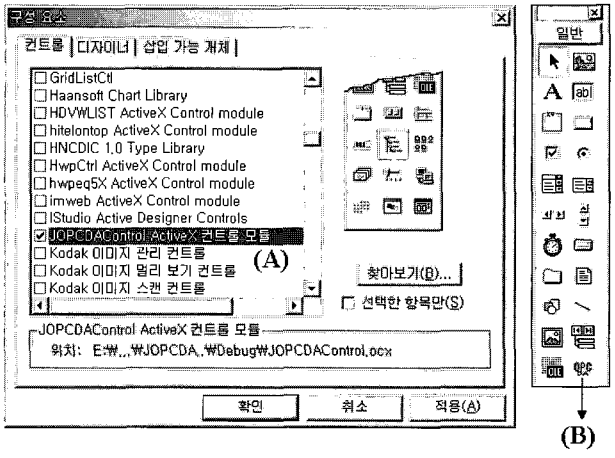


그림 9. 컴포넌트의 추가.
Fig. 9. Add component.

1. OPC server의 선정

OPC client를 제작하기 위해서는 우선 연결 대상이 되는 OPC server를 선정해야 한다. 연결 대상이 되는 OPC server는 OPC foundation에서 정의한 인터페이스를 모두 지원해야 한다. 이를 검증하기 위해 OPC foundation에서는 제공하는 compliance tool을 이용하여 테스트를 수행하였다. 테스트는 OPC foundation의 OPC data access 2.05a sample server, Softing사의 softing OPC toolbox demo server, technosoftware AG사의 OCS toolkit DA sample server v3.3, 그리고 Merz사의 FastOPCS imulator에 대하여 수행되었다. 이 중 OPC Foundation의 OPC Server만이 테스트를 완벽히 통과하였다. 그림 8은 OPC foundation의 server의 테스트 결과를 보여준다.

2. OPC Client의 디자인

연결할 server를 정했으면 이제 OPC client를 디자인한다. 우선 visual basic에서 본 연구에서 구현한 컴포넌트를 사용할 수 있도록 프로젝트 구성요소에 컴포넌트를 추가한다. 그림 9는 프로젝트 구성요소에 컴포넌트를 추가하는 모습을 보여준다. (A)는 컴포넌트 구성요소에 본 연구에서 구현한 컴포넌트를 추가하는 모습을 나타내고, (B)는 visual basic의 툴바에 컴포넌트가 추가된 모습을 보여준다.

컴포넌트가 추가되었으면 이제 추가된 컴포넌트와 visual basic의 기본 컨트롤들을 이용해 OPC client를 디자인한다. 그림 10은 디자인한 OPC client를 보여준다. (A)가 본 연구에서 구현하는 컴포넌트이다. OPC client는 OPC server에 접속 및

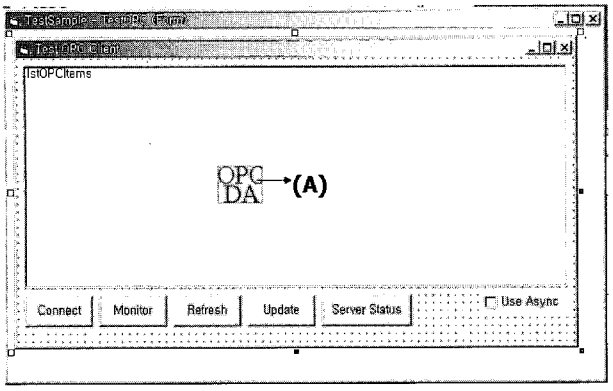


그림 10. OPC client의 디자인.
Fig. 10. Design of OPC client.

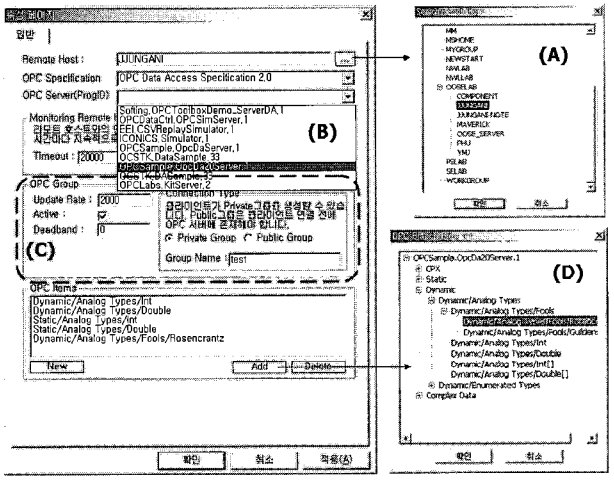


그림 11. 컴포넌트의 속성 설정.
Fig. 11. Set up properties of component.

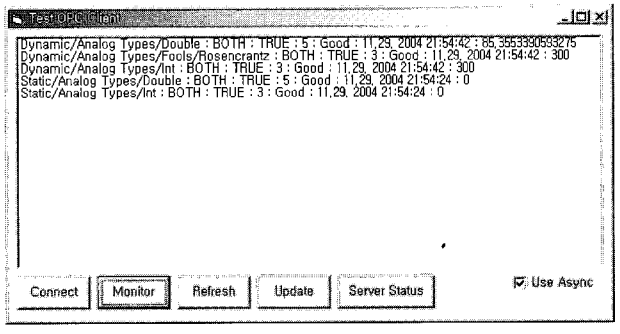


그림 12. OPC client의 실행.
Fig. 12. Execution of OPC client.

접속 해제, 데이터의 수집과 수정, OPC server의 상태 확인을 수행할 수 있도록 디자인 되었다.

3. 컴포넌트의 속성 설정

OPC client의 디자인이 끝나면 컴포넌트의 속성 설정을 통해 OPC server와의 연결에 필요한 정보들을 설정한다.

그림 11은 OPC server와의 연결에 필요한 정보들을 컴포넌트에서 제공하는 GUI환경에서 설정하는 화면을 보여준다.

우선 가장 먼저 설정할 속성은 연결할 OPC server가 존재하는 원격 호스트를 선택하는 것이다. (A)는 현재 네트워크의 원격 호스트들을 검색하여 보여주는 화면으로 사용자가 쉽게 OPC server가 존재하는 원격 호스트를 선택할 수 있도록 하고 있다. 원격 호스트를 선택하고 난 후에는 연결한 OPC server를 선택하도록 한다. (B)는 선택한 원격 호스트에 존재하는 OPC server의 리스트를 보여주고 있다. 앞선 테스트 결과 OPC foundation의 OPC data access 2.05a sample server를 연결 대상 서버로 지정했으므로 해당 server를 선택한다. 연결할 server를 선택한 뒤에는 서버에 생성할 그룹에 대한 속성을 설정한다. (C)는 지정한 OPC server에 생성될 그룹의 속성을 설정하는 부분이다. 기본적으로 client는 하나의 private 그룹을 생성하여 사용하지만 기존에 존재하는 public 그룹을 선택하여 사용할 수도 있다. 그룹에 대한 설정이 끝났다면 이제 OPC server의 사용할 아이템들을 선택한다. (D)는 지정한 OPC server의 address space를 검색해 보여주는 부분으로 GUI 기반으로 검색이 가능하여 쉽게 아이템을 추가, 삭제할 수 있다.

4. OPC Client의 실행

컴포넌트의 속성 설정이 완료되었으면 OPC client를 실행해본다. 그림 12는 제작한 OPC client를 실행한 모습을 보여준다.

실행결과 제작된 OPC client는 OPC server에 정상적으로 연결되어 서버의 데이터에 접근 및 조작을 수행하는 것이 가능하였으며, 구현한 컴포넌트를 통해 쉽게 OPC client를 제작할 수 있었다.

V. 결론 및 향후 연구과제

본 연구에서 구현하는 ActiveX 기반의 컴포넌트는 OPC client의 개발 시 COM/DCOM에 대한 전문적인 지식이나 프로그래밍에 대한 지식이 부족하더라도 쉽게 OPC server에 접근하여 데이터를 조작할 수 있는 방법을 제공한다.

본 연구에서 구현하는 컴포넌트의 장점을 정리하면 다음과 같다.

- GUI 환경을 제공함으로써 OPC server와의 통신에 필요한 속성을 쉽게 설정할 수 있다.
- OPC server와 통신하기 위하여 필요한 COM/DCOM 통신을 개발자가 직접 수행하지 않기 때문에 OPC client의 개발과 유지보수가 쉬워진다.
- OPC server와의 통신을 하기 위한 복잡한 과정을 대신 수행함으로써 OPC 시스템의 구조에 대한 이해가 부족하더라도 쉽게 OPC client 어플리케이션을 구현할 수 있다.
- ActiveX 컨트롤의 형태로 제공되므로 간단한 교육만으로도 쉽게 어플리케이션을 작성할 수 있다.
- OPC client의 쉽고 빠른 개발 및 유지 보수를 지원함으로써 OPC client의 개발 및 유지 보수 비용을 줄일 수 있다.

향후에는 현재 OPC data access 표준만 지원하고 있는 컴포넌트를 OPC의 다른 명세들(XML data access, historical data access, alarm & event, 등)도 지원할 수 있도록 하는 방법에 대한 연구와 다수의 컨트롤이나 컴포넌트와 연동할 수 있도록 커넥션 포인트 컨테이너에 대한 연구가 필요하다. 그리고 각

각의 명세를 지원하는 컴포넌트들을 통합하는 OPC client 컴포넌트에 대한 연구도 진행될 것이다.

참고문헌

[1] OPC Foundation, OLE for Process Congrol Standard, "http://www.opcfoundation.org"

[2] Component Object Model Specification, "http://www.microsoft.com/"

[3] ActiveX controls, "http://msdn.microsoft.com/"

[4] M. Riedl and M. Thron and T. Hadlich, "Drive server-significantly reduce in engineering expense," *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, Volume: 1, 29 Nov.-2 Dec. 2001 Pages: 285-288 vol. 1.

[5] M. Janke, "OPC-plug and play integration to legacy systems," *Pulp and Paper Industry Technical Conference, 2000. Conference Record of 2000 Annual, 19-23 June 2000* Pages: 68-72.

[6] V. Kapsalis and K. Charatsis and M. Georgoudakis and G. Papadopoulos, "Architecture for Web-based services integration," *Industrial Electronics Society, 2003. IECON '03. The 29th Annual Conference of the IEEE, Volume: 1, 2-6 Nov. 2003* Pages : 866-871 vol. 1.

[7] V. Kapsalis and K. Charatsis and M. Georgoudakis and G. Papadopoulos, "OPC-SMS: a wireless gateway to OPC-based data sources," *Computer Standards & Interfaces, Elsevier Science BV*, vol. 24/5, pp. 437-451, November 2002.

[8] Merz Software, Easy OPC-DA, "http://www.merz-sw.com/opc/cop3_easyopc.php3"

[9] Technosoftware AG, TsOPC Framework, "http://www.tsopc.com/modules/solutions/index.html"

[10] Eldridge Engineering Inc, OPC for Visual Basic, "http://www.opcsystems.com/OPCforVB.htm"

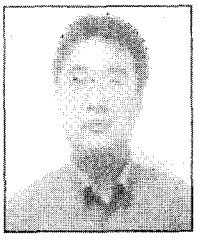
[11] Softing, OPC Toolbox ActiveX, http://softing.com/en/communications/products/opc/tools/activex.htm

김 종 환

2003년 울산대학교 컴퓨터정보통신공학부 졸업. 2005년 동 대학원 석사. 2005년~현재 (주) 드림위즈. 관심분야는 컴포넌트 프로그래밍, 자동화 시스템.

이 명 재

1987년 서울대학교 전산과학과 졸업. 1989년 동 대학원 석사. 1995년 동 대학원 박사. 1996년~현재 울산대학교 컴퓨터정보통신공학부 부교수. 관심분야는 소프트웨어 공학, 공장 자동화.



심 민 석

1999년 울산대학교 전자계산학과 졸업. 2002년 동 대학원 석사. 2002~현재 울산대학교 컴퓨터정보통신공학과 박사과정 재학 중. 관심분야는 소프트웨어 공학, 컴포넌트 프로그래밍, 공장 자동화.