OPC UA 전용 클라이언트 개발을 위한 C# 컨트롤의 설계 및 구현

신준철○ 유대승 이명재
울산대학교 컴퓨터정보통신공학부
ducksjc@hotmail.com, ooseyda@mail.ulsan.ac.kr, ymj@mail.ulsan.ac.kr

Design and Implementation of the C# Control for Development of an OPC UA Specific Client

Joon-Choul Shin○ Dea-Seung Yoo Myeong-Jae Yi
School of Computer Engineering and Information Technology University of Ulsan

요 약

프로세스 컨트롤 장비를 제어하기 위한 산업규격인 OPC는 새로운 규격으로 UA를 정의하여 그 개발 효율성 증대의 가능성을 열었다. 그러나 UA 규격은 방대하고 복잡한 구조를 가지고 있어 본 연구에서 C# 컨트롤을 통해 개선된 CBD환경을 구축한다. 이 환경에서는 GUI를 통해 효율적으로 OPC UA 전용 클라이언트를 개발할 수 있다.

기존의 UA SDK와 UA 범용 클라이언트 소스를 재사용하고 C#의 특성을 살려 사용자 컨트롤을 만들어 디자인타임에서의 GUI 개발환경을 지원한다. 그리하여 개발을 편리하게하고 효율적으로 유지보수가 가능하게 한다.

1. 서 론

프로세스 컨트롤 분야에 산업 규격으로 떠오른OPC는 복잡한 규격과 COM/DCOM 기반으로 구축으로 인해 이해하기 힘들고 개발의 불편함을 가져왔다 그리하여 새로이 UA(OPC Unified Architecture)[1] 규격을 만들어 C#을 기반으로 하고 사용자 컨트롤 개발이 용이하게 되었으며, SDK 와 샘플을 통하여 근본이 되는 소스를 재사용하기 용이하도록 지원하고 있다 그러나 UA 규격 자체가 방대하고 복잡하여 여전히 사용자에게 어려움을 느끼게 하고 있다.

이런 개발의 어려움을 개선하기 위하여 CBD(Compoment Based Development) 환경을 구축할 수 있다. 근본적으로 OPC UA는 C#을 지원하고, C#의 특징을 살려 CBD 환경을 기반으로 하고 있다 그러나 UA SDK에서 지원하는 샘플과 범용 클라이언트의 소스들은 이해하기 쉽지 않으며 가장 밑바닥의 라이브러리 형태를 취하고 있어 개발의 어려움을 가지게 한다 그래서 이 SDK를 사용하여 한 단계 위의CBD 환경을 구축함으로 좀 더 효율적인 개발을 지원할 것이다

본 논문에서는 OPC UA 전용 클라이언트 제작을 위하여 특별히 만들어지는 C# 컨트롤을 설계하고 구현한다 이 컨트롤은 OPC UA에 대한 전문적인 지식 없이도 간단한 GUI 프로그래밍 방식을 통하여UA 서버와의 접속과 필요한 노드들을 찾고 접촉하여 데이터를 수집하고 주기적으로 갱신하는 등의 일련의 과정을 간단하게 만들어 준다.

따라서 다양한 요구사항에 맞추어 변화하는 전용 클라이언트의 제작의 효율성과 유지보수를 극대화 시킬 수 있다. 또한 본 연구의 컨트롤 자체 또한 재사용성의 여부를 남김으로서 더욱 뛰어나고 효율적인 컨트롤 제작의 가능성을 가지고 있다

본 논문의 구성은 다음과 같다 이어지는 2장에서는 이전에 연구되었던 관련된 기술을 알아본다 제 3장에서는 본 연구에서 구현하는OPC UA 컨트롤의 전체 디자인과 내부 구성을 알아본다 마지막 제 4장 에서는 결론 및 향후 연구과제에 대해서 논의하면서 본 연구에서 미흡했던 부분과 앞으로 개발 가능성이 있는 부분에 대해서 알아본다.

2. 관련연구

OPC는 1996년 발표된 이후로 다양한 연구가 이루어졌으며 최근에 OPC UA가 발표되어 새로운 연구들이 진행되고 있다. 국내에서는 UA에 맞춘 새로운 연구는 미미한 상태이며 국외에서는 관련된 제품이 상품화 되고 있는 단계에 도달해 있다.

OPC 규격에 맞추어 컴포넌트를 만들고 ActiveX의 형태로 지원하는 연구[2]가 있다. OPC에 대한 이해가 부족하더라도 쉽게 자신만의 클라이언트를 제작하고 유지보수를 유용하게 하는 컴포넌트를 지원하는 것이다 그러나 UA의 지원은 없으며 OPC-DA 규격만을 일부 지원하는 제한을 가지고 있다

UA 규격이 정의되어가는 시점에서 SOAP를 사용하여 UA와 흡사한 시스템을 구축하여 댐의 발전소를 원격 제어하는 연구[3]가 있었다. UA 규격이 나오기 이전에 이미 실험으로 UA의 가능성이 연구되었던 것이다

OPC UA의 규격을 더욱 효율적으로 이용하여 하드웨

어 디바이스의 정보를 표현하는 추가적인 기술에 대한 연구[4]가 있다. 모든 하드웨어에 대해 OPA UA규격을 적용하여 디바이스의 타입과 정보를 좀 더 효율적으로 정리하고 표현하는 기술에 대한 시도다.

OPC 규격을 사용한 GUI를 지원하는 상품은 몇가지 있다. Softing사의 OPC Toolbox[5]가 그 중에 하나다. 그러나 여전히 Softing사에서는 UA에 대한 지원은 준비 단계에 있다.

OPC UA를 컨트롤로 만드는 연구는 몇몇에서 진행되고 있고 만들어진 것이 있으나 UA 규격의 일부만을 지원할 뿐 모두 한계가 있다.

3. OPC UA 전용 클라이언트 구현

클라이언트는 크게 2가지로 구분된다. 모든 서버에 접속이 가능하며 브라우징이 가능한 범용 클라이언또브라우저)와 특정한 용도를 가지며 최종 개인 사용자를 위한 전용 클라이언트로 나뉜다. 브라우저의 경우 훌륭한 하나의 브라우저가 만들어 짐으로서 모든 사용자가 사용할 수 있지만 전용 클라이언트는 사용자 별로 만들게 되고 그 질이 떨어지기 쉽다.

다양하게 만들어야하고 상황이 바뀌면 다시 만들게 되는 전용 클라이언트는 그 개발 과정이 쉽고 간편하며 유지보수 또한 쉬워야 한다. 그러나 OPC UA의 규격의 복잡함으로 인해 프로그래밍 지식이 낮은 개인 사용자들은 개발의 어려움을 느끼게 된다.

본 연구는 이 전용 클라이언트 개발의 효율성을 위해 브라우저의 기본적인 기능을 내포하고GUI를 통해 개발을 쉽게 해주는 C# 컨트롤을 개발한다.

3.1 OPC UA 범용 클라이언트 재사용

일반 브라우저는 모든 서버에 접속이 가능하며 데이터의 종류에 상관없이 읽고 구독(subscribe)을 할 수 있도록 만들어지게 된다. 모든 규격을 지원하고 방대한 기능을 가지고 있다. 그 대신에 인터페이스가 복잡해지고 특정 용도에 맞추어서는 더욱 비효율적인 인터페이스를 가짐으로서 전용 브라우저의 필요성이 생겨난다.

내부적으로 이 클라이언트는 모든 복잡한OPC UA 규격을 지키며, 규격에서 지원하는 모든 기능이 구현되어 있다. 이 미리 만들어진 브라우저의 소스를 재사용하여 전용 클라이언트를 개발하는데 사용할 수 있다.

한층 더 높은 개발의 효율을 위하여 브라우저의 소스를 사용하여 전용 클라이언트를 위한 컨틀롤을 만드는 것이 본 연구의 주제다. 서버에 접속하기 위해 인증을 거치고, 노드들의 구조를 파악하는 브라우징 기능을 컨트롤에 맞추어 수정할 수 있다.

3.1 컨트롤 전체 구조

본 연구에서 구현하는 컨트롤은 Microsoft Visual Studio 2005 환경에서 C#을 이용하여 구현되었다.

전체 디자인은 기존의 소스를 재사용하며 본 연구의 결과물 또한 재사용이 용이하고 Visual Studio 2005에서 지원하는 디자인타임의 일반적인 방식을 지킴으로 사

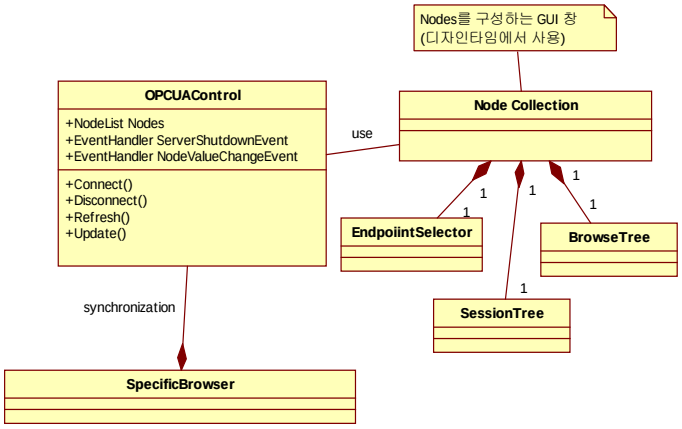

그림 1 Control Class Diagram

용이 용이하도록 함에 중점을 두었다. 또한 전체적인 클래스 구조가 응집성은 높고 결합도가 낮도록 디자인 되었다.

그림 1의 OPCUAControl 은 컨트롤 클래스 그 자체를 나타낸다. 최종 개발자는 이 OPCUAControl 하나만을 이용하여 개발하게 된다. Node Collection 은 GUI를 지원하는 클래스로 범용 클라이언트로서 일부 기능과 특정 노드를 수집 정리하는 기능으로 나뉜다. 가장 우측의 EndpointSelector를 포함하여 3개의 클래스는 Node Collection을 구성하는 세부 컨트롤이다. 각각 서버와의 접속, 세션 관리, 노드 브라우징을 지원한다.

우측의 3개의 클래스는 기존의 SDK에서 지원하는 샘플 클라이언트의 소스를 재사용한 것으로 각각이 또 하나의 컨트롤이다.

3.2 디자인 타임 GUI 지원

CBD의 장점 중 하나는 각 컴포넌트를 가져와 설정하는데 있어서 GUI를 지원할 수 있다는 것이다. 본 연구에서 만들어지는 컨트롤 또한 컴포넌트의 하나로 최종 개발자가 사용하는데 있어서GUI를 지원한다.

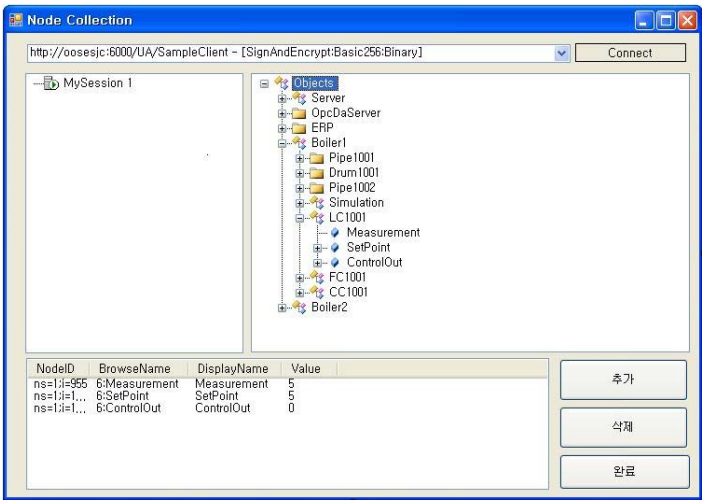

그림 2 Node Collection 대화 상자

본 연구에서 가장 추구하는 바는 GUI를 통한 쉬운

OPC UA에 접근이다. 위 그림 2의 대화상자는 최종 개발자가 전용 클라이언트에서 사용하는 서버와 노드들을 선택하는 과정을 보여준다. 위 대화상자가 그림 1의 Node Collection 클래스이다. 이 클래스를 구성하는 각각의 컴포넌트 또한 다른 컨트롤에서 가져와 재사용 되고 있다. 좌측 상단의 Connect와, 상단의 http 주소줄이 EndpointSelector다. 좌측의 MySession을 표기하는 상자는 SessionTree 컨트롤이다. 우측에 많은 노드의 트리를 보여주고 있는 상자가 BrowseTree 이다.

기존의 컨트롤은 본래 OPC SDK의 샘플 클라이언트와 ClientControl에서 사용되는 것들이다. 각각의 컨트롤을 일부 수정하여 재사용함으로서 비교적 쉽게 OPCUAControl 을 GUI로 구성하고 있다.

OPC UA이 규격을 따르면 노드들의 구조가 항상 트리 형태를 취하지는 않는다. 하지만 대체로 트리 구조를 지향하여 흡사하게 보일 뿐 완전한 트리 구조가 아니다. 예를 들어 A노드에서의 자식노드가 B노드에서도 나타날 수 있다. 어떤 경우든 위의 트리 형태에서 원하는 노드를 찾을 수 있다. 이 가벼운 혼란은 최종 개발자에게 OPC UA에 대한 약간의 이해를 요구하나 그런 이해가 없다고 하여 큰 문제가 되지는 않는다. 이런 GUI의 형태를 다룸으로서 전용 클라이언트 개발자는 약간의 이해를 통해서 개발을 해나갈 수 있게 된다.

3.3 컨트롤의 메소드

본 연구에서 제작하는 컨트롤은 몇개의public 메소드와 속성을 가지고 있다. 각각의 메소드와 속성은 고유의 역할을 가지고 있으며 그 역할이 매우 분명하고 간단하다.

표 1 컨트롤이 지원하는 메소드

메소드	설명
Connect	서버와의 연결
Disconnect	서버와의 연결 해제
Refresh	데이터 다시 받기
Update	데이터 갱신하기

일반적으로 OPC UA 서버와의 연결은 매우 복잡한 루트를 거친다. Connect는 그 루트를 매우 간편하게 줄여준다. 본래 개발단계에서 GUI를 통해 서버와의 접속에 필요한 모든 루트를 미리 거치게 되고 Connect는 그 접속에 사용한 모든 데이터를 저장하여 호출시 그대로 수행하게 된다. 이런 데이터로는 유저 이름과 패스워드 Endpoint와 세션의 이름 등이 있다.

일부 Endpoint(OPC UA 서버의 위치)가 변경될 경우에 런타임에서 바로 수정이 가능하도록 하기 위하여 인자를 받는 Connect 함수를 지원할 필요가 있다. 이런 경우 EndpointSelector 컨트롤을 추가하여 최종 유저는 2개의 컨트롤을 사용하게 될 것이다 물론 복잡도가 올라가며 유저로부터 이해를 요구하게 된다

접속 종료는 Disconnect 메소드를 통하여 지원한다. Refresh는 Node Collection에서 지정한 노드들의 값을 다시 받는다. 그러나 EventHandler를 통하여 값이 변했을 경우 자동으로 이벤트가 발생하도록 사용자 메소드와 연결할 수 있다. Update 함수는 사용자가 노드의 값을 수정하기 위해 호출하는 함수로 OPC UA 서버에 노드를 수정하기 위한 복잡한 절차를 줄여준다.

3.4 컨트롤의 속성

디자인 타임에서 지정한 노드들을 저장하기 위하여 간단한 리스트 형태로 노드들의 저장소인Nodes를 속성으로 가지고 있다. 사용자는 단순히 Nodes를 읽음으로서 OPC UA 서버 노드의 값을 읽을 수 있다. 해당 데이터가 실시간으로 갱신되는 것은 아니지만 정기적인 구독을 통해 데이터를 갱신한다. 보통 1초에 한번정도 갱신하게 된다.

데이터를 갱신한다 하여도 그 값이 변하지 않은 경우 사용자 입장에서는 의미가 적다 그래서 데이터의 변화를 감지하여 Event를 일으키는 NodeValueChangeEvent를 지원한다. 이 EventHandler에 최종 개발자가 만든 메소드를 연결함으로서 간편하게 데이터의 변화를 감지할 수 있게 된다.

ServerShutdownEvent는 서버의 종료나 서버와의 비정상 적인 종료를 감지하면 발생하는 이벤트다.최종 개발자는 이 이벤트를 통하여 데이터 읽기를 중지하고 재접속을 시도하거나 프로그램 종료를 할 수 있게 된다.

표 2 컨트롤이 지원하는 속성

속성	설명
Nodes	지정한 노드들의 리스트
ServerShutdownEvent	서버 종료 이벤트
NodeValueChangeEvent	데이터 변화 이벤트

3.5 컨트롤을 이용한 전용 클라이언트 구현

본 연구에서 구현한 컨트롤을 이용하여 개별화된 클라이언트를 제작하는 일반적인 과정은 다음과 같다.

1. OPC UA 서버 Endpoint 찾기

접속하고자 하는 UA 서버의 위치가 되는 Endpoint를 찾아야 한다. 네트워크 상에 접속 가능한Endpoint를 정리하고 SecureChannel과 세션 설정 등의 번거로운 작업을 GUI를 통하여 해나간다.

2. 노드 정하기

서버를 구성하는 노드를 트리 구조로 보면서 개발하고자 하는 클라이언트가 필요로 하는 노드를 찾아서 지정한다. 지정한 노드들은 리스트 형태로 노드ID를 저장하여 런타임에서 읽어 들일 수 있도록 한다.

3. 전용 클라이언트 디자인

컨트롤의 설정을 마치면 전반적인 클라이언트를 디자인하고 컨트롤의 속성들을 사용하여 코드를 작성한다. 각각의 이벤트에 최종 개발자의 메소드를 연결하고 노드 값을 사용하여 화면에 출력하는 등의 작업을 한다.

4. 클라이언트의 실행

컨트롤의 설정과 사용하는 코드가 모두 완성되었다면 클라이언트를 실행할 수 있다. 그림 3은 간단하게 완성해본 클라이언트의 실행 장면이다. 본 클라이언트는 간단히 3개의 데이터를 지정하여Refresh한 모습이다.

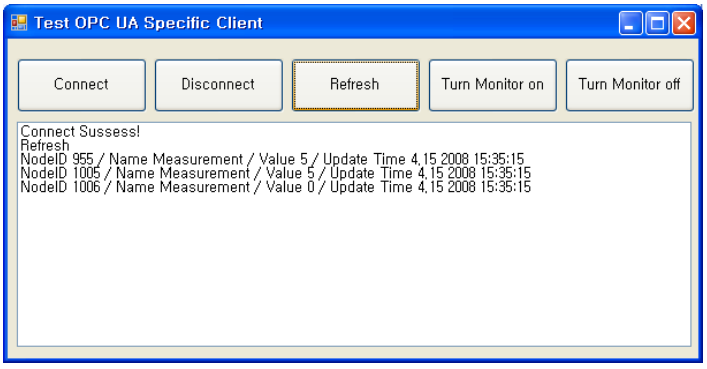

그림 3 클라이언트 실행 화면

4 결론 및 향후 연구과제

OPC UA 서버를 찾고, 노드를 찾아 읽고 갱신하는 등의 일련의 작업을 컨트롤을 사용하여 효율적으로 할 수 있게 됨으로 전용 클라이언트 제작을 용이하게 하고 유지보수를 효율적으로 할 수 있다. 특히 방대한 양의 UA 규격을 알지 못해도 명확하고 간단한 GUI 개발 환경을 통해 개발이 가능하다.

본 연구의 컨트롤이 가지는 장점을 정리하면 다음과 같다.

•GUI 환경을 구축함으로 OPC UA 서버와의 연결에 필요한 일련의 과정을 복잡한 프로그래밍 코드 없이 가능하게 한다.

•실제로 사용하는 노드만을 고르기 위해 노드 브라우저를 GUI로 지원하여 전용 클라이언트 개발을 용이하게 한다.

•UA에서 정의하는 Subscribe와 AddressSpace등의 전문 지식 없이도 컨트롤에서 public으로 지원하는 Nodes 속성을 읽음으로 간단하게 데이터에 접근한다.

•C# 컨트롤 형태의 구현이기 때문에 그 사용이 극히 단순하여 간단한 교육만으로도 쉽게 어플리케이션을 개발할 수 있다.

•다양한 컨트롤 프로세서 장비와 사용 목적에 따른 다양한 전용 클라이언트를 개발할 수 있도록 개발 및 유지보수 비용을 줄일 수 있다.

UA 규격은 다양한 방법으로 서버를 이용하지만 본 연구의 컨트롤은 그의 일부분만을 지원할 뿐이다. 향후에는 좀 더 사용자의 요구에 맞추어 UA 규격의 다양한 부분을 지원하면서 사용의 복잡함 증가를 최소화 시키는 방법에 대해 연구가 필요하다.

본 연구는 현재 오직 C#만 지원한다. 다중 플랫폼을 위하여 Java와 C++에서 본 연구의 컨트롤을 사용할 수 있는 다양한 방법을 연구하여 사용과 개발의 폭을 늘리도록 할 필요가 있다.

본 연구에서의 컨트롤은 최종적으로 사용자에게1개의 컨트롤만 보인다. 다양한 UA 규격을 지원함으로 단일 컨트롤로의 디자인은 오히려 사용의 불편함과 비효율성을 가질 수 있다. 향후 약간의 이해를 요구하지만 효율적인 클라이언트 개발을 위해 컨트롤의 분리를 연구하여 서버와의 접속, UA 메소드의 실행 등을 다루는 고급 개발자용의 컨트롤 연구가 진행될 것이다.

참 고 문 헌

[1] OPC Foundation, OPC Unified Architecture "Http://www.opcfoundation.org"

[2] 유대승, 효율적인 OPC Client 생성을 위한 ActiveX 기반 프레임 워크, 전력전자학술대회논문집 Pages 621~623(3 pages), Jul 2005

[3] Pasteur and Olivier Tuan Dang and Pierre-Etienne Delon, Feedback from SOAP based messaging protocol for power generation information system, "Control & Automation, 2007. MED '07. Mediterranean Conference", Page 1-5, 2007

[4] Thomas Hadlich, Providing device integration with OPC UA, "Industrial Informatics, 2006 IEEE International Conference on", Pages 263-268, Aug 2006

[5] Softing, OPC Toolbox, "http://www.softing.com/home/en/industrial-automation/products/opc/toolkits.php"