컴포넌트 오브젝트 모델을 응용한 계전 알고리즘 구현방법의 개선에 관한 연구

박 인 권 , 윤 남 선 , 안 복 신
LG 산전 전력 연구소

A Study on the Improvement of the Method for Implementing Protection Algorithm by using Component Object Model(COM)

Park in kwon, Yoon Nam Seon, An Bok Shin
LG Industrial Systems

Abstract - The complexity of newly developed protection algorithm and higher performance requested by the user makes the software embedded in the protective relay much harder to develop and maintain. The versatility of 32bit microprocessor and the availability of cheaper memory semiconductors introduced the fertile developing background for the protective relay developers. The use of component object model(COM) in the software developing process enables the developer to write much complex code in the easy and safe way and to maintain the code easily, too. And the aid of the COM library, the distributed computing environment will be expected to appear by the use of the COM programming model in the protective relay firmware program.

1. 서 론

근래 전력계통의 대규모화 및 복잡화로 말미암아 여러 새로운 계전 알고리즘이 개발되어 이용되고 있다. 과거 이러한 알고리즘을 실제 전력계통의 보호에 이용되는 디지털 계전기에 구현하기 위해서는 이용되는 프로세서의 계산능력의 부족과 이용 가능한 메모리의 제한으로 말미암아 대부분 어셈블러 등의 저 수준의 언어로 이러한 알고리즘을 구현하는 것이 보통 이었고 이는 기존에 작성된 소프트웨어의 재사용을 극히 어렵게 만들 뿐 아니라 사용자의 요구사항의 변경에 따른 소프트웨어의 유지보수에도 많은 어려움을 끼쳐왔다.

그러나 근래 이루어진 32Bit 마이크로 프로세서의 비약적인 발전 및 가격의 대폭적인 하락, 또한 반도체 메모리 가격의 전세계적인 폭락은 고기능의 하드웨어 및 소프트웨어의 개발을 가능하게 하여주는 밑바탕이 되어 주고 있으며 이러한 기능상의 발전은 기존의 디지털 계전기가 가지던 단순 계측 또는 보호기능을 포함함을 물론 네트웍등 고수준의 통신 기능 및 인공지능 기법을 응용한 판단기능까지 구현된 복잡한 보호계전기의 등장도 가능케 하고 있다. 또한 최근의 마이크로 프로세서의 개발동향은 개발자로 하여금 더욱 친숙하고 용이한 환경에서 작업할 수 있도록 배려할 뿐만 아니라 최적 화된 고급언어(C++등)컴파일러의 제공은 어셈블리로 작성된 프로그램의 효율에 버금가는 성능의 목적코드를 제공하여 준다.

그러나 이미 대형 소프트웨어에서 등장하는 바와 같이 발전되는 알고리즘의 난해함에 더불어 알고리즘의 구현물인 프로그램 파일의 복잡화는 개발된 소프트웨어의 유지 보수 및 개선작업을 더욱 더 어렵게 하고 있는 것도 사실이다.

본 논문에서는 객체지향 기법에 기반을 둔 COM(Component Object Modeling) 기법을 응용하여 전자 회로에서 마치 각기 특정 기능을 수행하는 IC들을 조합하여 전체 기능을 수행할 시스템을 구성하듯 소프트웨어 컴포넌트를 이용하는 것을 가능하게 하여주는 구현방법을 취함으로서 이미 개발된 소프트웨어의 유지보수 및 재사용 성을 높이는 방법을 제시하고자 하였다. 또한 기반이 되는 운영체제에서 제공되는 오브젝트의 인터페이스를 통한 오브젝트간의 통신기능을 통하여 프로토콜 레벨에서의 통신이 아닌 보호계전기 수준에서의 분산처리 가능성을 탐색하여 보았다.

2. 본 론

개인용 컴퓨터의 급속한 발전과 인터넷으로 대표되는 네트웍 기술의 광범위한 이용은 다양한 프로그램의 시공간에 제약을 받지 않는 이용을 가능케 하였다. 또한 이러한 효용을 인식하고 이를 사용하여 생산성의 비약적인 발전을 증가시킬 수 있었다.

현대 사회는 더욱 새롭고 다양한 분야의 컴퓨터를 사용하기를 바라며 교육, 의료, 생산, 의료 출판 등에 대한 새로운 요구가 끊임없이 생겨나고 있다. 현재 국내의 소프트웨어 시장만 봐도 매년 30%이상의 고속성장을 거듭하고 있다. 컴퓨터 분야의 기술발전은 과거보다 더 크고 복잡한 소프트웨어를 개발할 수 있게 하고 소프트웨어의 발전은 더 많은 소프트웨어의 개발을 이끌어낸다.

이러한 발전은 전력분야도 예외가 아니어서 EMS, SCADA를 비롯한 상위시스템 뿐만 아니라 개별 디지털 보호 계전기 등의 IED(Intelligence Electronic Device)에 이르기까지 더욱 고기 능의 소프트웨어를 탑재하고 더욱 복잡한 알고리즘을 수행하는 제품이 매년 시장에 쏟아져 나오고 있다.

2.1 컴포넌트 오브젝트 모델(COM)

위에서 살펴 본 것처럼 보호 계전기에 구현되는 소프트웨어가 복잡하고 그 규모가 커지게 될수록 하나의 어플리케이션에 모든 기능을 포함시키는 것보다는 여러 개의 단위 어플리케이션 즉, 컴포넌트로 분할하는 것이 재사용성과 유지보수의 측면등 여러 가지 면에서 더욱 효율적임은 자명한 사실이다. 현재 개인용 컴퓨터 및 고기능의 실시간 시스템에서 이용되는 단위 어플리케이션간의 통신방법은 다음과 같다. 1) 동적 링크 라이브러리(Dynamic Linking Library), 2) 동적 데이터 교환(DDE : 윈도우즈 운영체제에 국한된 데이터 교환방법), 3) 원격 프로시저 호출(Remote Procedure Call), 4) 시스템 호출(System Call). 하지만 이들 통신 메커니즘은 서로 상이할 뿐만 아니라 이용되는 컴포넌트의 버전관리도 용이한 일은 아니다. 또한 단위 어플리 케이션이 서로 상이한 플랫폼 상에서 상이한 언어로 작성되었다면 단위 어플리케이션 사이의 케뮤니 케이션조차 어려운 경우가 발생하게 된다.

이러한 문제점들을 해결하기 위해 등장한 것이 컴포넌트 오브젝트 모델(COM, Component Object Model)

이다. 컴포넌트 오브젝트 모델은 이들 단위 어플리케이션, 즉 컴포넌트들이 커뮤니케이션 하는 방법을 구현한 사양(Specification)이다.

COM 컴포넌트는 다음과 같은 사양을 만족해야 한다.

1) COM 컴포넌트는 완전히 언어 독립적 이여야 한다.

COM 컴포넌트는 C 와 같은 절차적 언어나 C++과 같은 객체지향적 언어, 또한 마이크로 소프트 사의 비주얼 베이직 같은 매크로 언어등 어떤 언어로도 개발할 수 있다.

2) COM 컴포넌트는 이진(Binary)형태로 제공되어야 한다.

3) COM 컴포넌트는 하위 버전과의 호환성을 제공하여야 한다.

즉, COM 컴포넌트가 업그레이드되어도 이전 버전을 사용하는 클라이언트 어플리케이션이 사용할 수 있도록 호환성을 제공하여야 한다.

4) COM 컴포넌트는 위치 투명성(Location Transparency)를 제공하여야 한다.

즉, COM 컴포넌트가 어느 위치에 있든, 예를 들어 클라이언트 어플리케이션과 같은 어드레스 스페이스를 공유하는 영역에 있든(인 프로세스 서버), 클라이언트 어플리케이션이 실행되는 같은 시스템 상의 다른 어드레스 스페이스 상에 존재하든(로컬 서버), 또한 네트워크상의 서로 다른 시스템 상에 있든(리모트 서버) 상관없이 서로 같은 방법으로 접근될 수 있어야 한다.

모든 COM 컴포넌트는 하나이상의 인터페이스(Interface)를 지원하며 각 인터페이스는 하나 이상의 메소드(Method)로 구성된다. 메소드란 어떤 특정 행위를 수행하는 함수 또는 프로시저 이며 COM 컴포넌트를 사용하는 클라이언트 어플리케이션에 의하여 호출될 수 있다. 여기서 COM 컴포넌트를 통하여 서비스를 제공하는 어플리케이션이나 서비스를 제공하는 COM 컴포넌트 자체를 서버라 하며 COM 컴포넌트의 서비스를 이용하는 어플리케이션을 클라이언트라 한다.

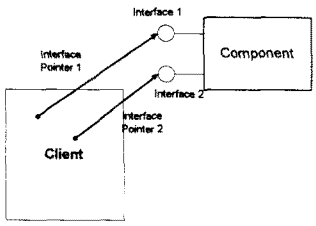

그림 2.1 COM 컴포넌트 인터페이스와 클라이언트

COM은 C++ 과 같은 언어가 아니며 따라서 어떠한 언어로든지 작성될 수 있다. 그러나 COM은 오브젝트 개념을 그 기초로 하고 있으므로 C++이나 자바 같은 객체지향언어로 작성하는 것이 편리하다. COM은 언어 독립적인 컴포넌트 라이브러리를 개발할 수 있는 구현방법을 제공하는 것뿐이며 실제적인 구현코드를 제공하는 것은 아니다. COM은 단지 사양(Specification)일 뿐이지만 실제적으로 사양 이사의 것을 제공한다. COM은 모든 클라이언트 애플리케이션과 컴포넌트에서 유용하게 사용될 수 있는 컴포넌트 관리 서비스를 제공하는 COM라이브러리를 제공한다. COM 라이브러리는 COM에서 가장 중요하다고 여겨지는 작업을 모든 컴포넌트에서 같은 방법으로 처리하기 위한 Co로 시작되는 API(Application Programming Interface)를 제공한다. 마이크로 소프트 사의 윈도우즈 운영 체제의 경우에는 Compobj.dll 또는 Ole32.dll의 형태로 제공된다.

2.2 인터페이스(Interface)

인터페이스란 COM 컴포넌트와 해당 컴포넌트를 사용하는 클라이언트 사이의 계약이다. COM 컴포넌트는 계약서(인터페이스)에 명시된 대로 서비스(메소드)를 제공한다는 것을 보장하고 클라이언트는 계약서(인터페이스)에 명시된 대로 안전하게 COM 컴포넌트의 서비스를 이용해야 한다. 일반적으로 계약은 계약당사자간의 합의가 없으면 파기될 수 없다. 마찬가지로 어떤 클라이언트에서 COM 컴포넌트의 인터페이스를 사용하고 있다면 일방적으로 해당 인터페이스를 수정할 수 없다.

프로그램 구현상 인터페이스는 C++과 같은 객체지향 언어에서의 순수 가상함수(Pure Virtual Function)만을 멤버로 포함하는 추상 클래스(Abstract Class)로 표현된다. 즉, COM은 모든 COM 컴포넌트의 인터페이스가 메모리 상에서 가상함수 테이블(Virtual Function Table)의 구조를 갖도록 규정한다. 그러나 인터페이스 구조를 꼭 객체지향적인 언어로서만 표현할 수 있는 것은 아니며 어떤 언어로 구현하든 가상함수 테이블 형태의 구조를 메모리 상에 표현 할 수 있으면 된다. 추상 클래스로 인터페이스를 정의하는 것은 이러한 메모리 구조를 정의하는 것이 되며 이 추상 클래스는 파생 클래스에서 구현될 때까지는 실제 메모리는 할당되지 않는다. 실제 파생 클래스에 메모리가 할당될 때의 구조는 다음 그림과 같다.

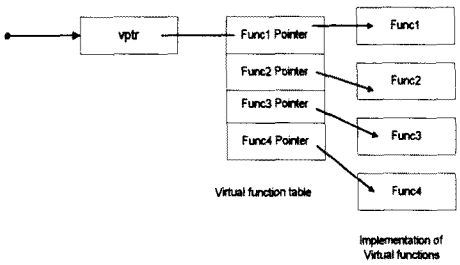

그림 2.2 인터페이스 메모리 구조

위 그림에서 볼 수 있는 바와 같이 인터페이스에 의해 구현되는 메모리 구조는 두부분으로 나뉘어 진다. 오른쪽의 가상함수 테이블은 가상함수를 구현함 함수의 포인터를 저장하는 배열이며 그림 왼쪽의 포인터 vptr은 가상함수 테이블을 가리키는 포인터인 가상함수 테이블 포인터가 위치한다. 이 가상함수 테이블 포인터를 인터페이스 포인터라고 하며 일단 인터페이스 포인터를 획득한 클라이언트 에서는 가상함수 테이블을 통하여 멤버함수 즉, 메소드를 호출할 수 있게 된다. 결국 COM인터페이스의 메모리 구조는 C++컴파일러가 추상 클래스에 대하여 생성한 메모리 구조와 완전히 일치하게 됨을 알 수 있으며 C++이 가장 자연스럽게 COM 을 구현할 수 있는 언어라는 것을 알 수 있다.

2.3 컴포넌트 오브젝트 모델을 응용한 계전요소 컴포넌트의 구현 및 컨테이너를 통한 테스트

컴포넌트 오브젝트 모델을 응용하여 프로그래밍을 하기 위해서는 상당한 수준의 객체지향언어에 대한 지식을 기반으로 한 컴포넌트 오브젝트 모델에 대한 지식을 필요로 한다. 또한 작성된 컴포넌트 오브젝트 모델 프로그

램의 정확한 동작을 보증하기 위해서는 여러 준수하여야 할 규칙이 있다. 예를 들면, 컴포넌트 오브젝트 모델 오브젝트를 생성하기 위해서는 반드시 CoGetClassObject나 CoCreateInstance 를 호출하여야 한다. 또한 이러한 과정을 거친 후 Addref나 Release 를 적절히 호출하고 호출한 컴포넌트가 정확히 동작하는지를 레퍼런스 카운트를 통하여 항시 확인하여야 한다. 이러한 사용법은 new 나 delete를 사용하는 것보다 훨씬 복잡하다. 또한 클래스 팩토리 등을 코딩해 주어야 하며 만들어진 오브젝트를 등록해 주는 스크립트도 작성되어야 한다. 그러나 이들 코드는 대부분의 컴포넌트 생성 시에 거의 동일하게 요구되는 과정이기 때문에 정형화가 용이하며 현재 여러 애플리케이션 프레임웍에서 이들 코드의 자동 생성기능 및 유지관리 기능을 지원하고 있다. 여기서는 마이크로 소프트 사의 Visual C++내에서 제공되는 C++ Template Class Library 인 ATL 2.1을 사용하여 기본적인 계전요소중의 하나인 DC Offset filter를 설계/구현하고 작성된 컴포넌트를 테스트 하기 위한 컨테이너 애플리케이션도 또한 작성하였다. ATL은 C++의 장점 중 하나인 템플릿(Template)클래스기능을 이용하여 최대한 작고 가벼운 COM컴포넌트를 제작할 수 있도록 지원한다. 템플릿은 다른 클래스 라이브러리와는 달리 다른 모듈과의 링크가 필요 없으며 컴파일러는 프리 프로세서에서 매크로를 확장하는 방법으로 컴파일 시간에 템플릿 클래스를 생성하게 된다. 결국 COM컴포넌트 내에는 실제로 사용될 템플릿 클래스만 포함되게 되며 또한 템플릿의 멤버 함수를 프로그래머가 원하는 대로 오버라이드 하여 사용할 수 있기 때문에 훨씬 유연한 프로그래밍이 가능하다. 다음은 작성된 COM컴포넌트의 인터페이스 다이어 그램 및 클래스 다이어그램 이다.

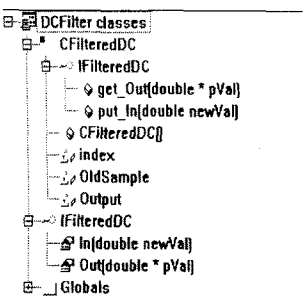

그림 2.3 인터페이스 다이어그램 및 클래스 다이어그램

다음 코드 부분은 작성된 컴포넌트를 이용하기 위한 인터페이스 포인터 및 인터페이스 포인터를 통한 프로퍼티의 이용예 이다.

```
#import ".\DCfilter.dll"no_namespace

// Interface Pointer
IFilteredDCPtr m_ptr;

for(i = 0; i < m_nItemCount; I++)
{

        m_ptr->put_In(m_dblVoltageA[i]);

        m_ptr->get_Out(&m_dblVoltageA_Out[i]);

}
```

다음은 위 컴포넌트 및 컨테이너를 이용하여 실제 정현파를 DC 필터링하여 얻은 결과이다.

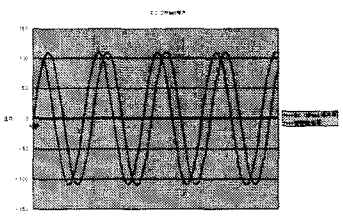

그림 2.4 컴포넌트 및 컨테이너의 이용

3. 결 론

보호계전 알고리즘의 복잡화 및 보호 계전기의 고기능화와 더불어 32비트 마이크로 프로세서의 광범위한 이용과 메모리 가격의 급격한 하락은 보호계전기에 내장되는 소프트웨어의 고기능화 및 복잡화를 필연적으로 수반하며 컴포넌트 오브젝트 모델을 이용한 계전기 알고리즘의 구현은 다양한 사용자의 요구에 쉽게 부응할 수 있고 유지 보수가 용이한 소프트웨어의 개발을 가능케 한다. 또한 근래 진행되고 있는 실시간 운영체제(RealTime Operation System)의 발전은 이러한 고급 프로그래밍 환경을 쉽게 하드웨어 상에서 이루어 질 수 있도록 도와주며 하위레벨의 구현방법에 독립적인 소프트웨어의 구현을 가능케 해 준다. 또한 이러한 운영체제에 탑재될 컴포넌트 오브젝트 라이브러리는 구현된 오브젝트간의 통신에 있어서 위치의 투명성(Location Transparency)을 보장하여 프로토콜에 독립적인 IED(Intelligent Electronical Device)의 출현을 가능케 할 것이다.

[참 고 문 헌]

[1] Dale Rogerson, "Inside COM", MS Press, 1997
[2] 전병선, "ActiveX 프로그래밍 가이드", 파워북, 1997
[3] 최은만, "소프트웨어 공학", 정익사, 1997
[4] Microsoft System Journal, June, 1997