소프트웨어 아키텍처를 이용한 IED용 PC 소프트웨어 개발

권효철*
(주)효성 중공업연구소

Development of PC Software for IED using software architectures

Hyo-Chul Kwon*
Hyosung Power & Industrial Systems R&D Center

Abstract - 최근 변전소용 IED는 IEC61850의 적용을 기점으로 하여 기술적으로 더 많은 발전을 거듭하고 있다. IED에 더 다양한 기술을 포함하면서 안정적이고 확실한 동작을 요구하는 것은 기본이고, 사용자 위주의 Interface와 편리한 관리 도구도 요구되어 진다. 이에 따라 IED를 PC로 제어하고 관리하기 위한 소프트웨어 또한 기술적으로 발전해야 하는 상황이다. 소프트웨어는 기본적으로 실시간 관리와 프로젝트 관리 그리고 다수의 IED지원 등의 기능을 가져야 하며 이를 구현하기 위하여 여러 가지 방식의 아키텍처 패턴이 적용될 수 있다. 본 논문에서는 IED를 관리하는 소프트웨어에 적용될 수 있는 아키텍처를 제시함으로써 소프트웨어 구현에 대한 방향을 모색해 보았다.

1. 서 론

IED를 관리하기 위한 소프트웨어는 세계 선두의 업체들 제외한 많은 업체들에서 IED구매 시 무료로 제공하고 있다. 아직까지 우리나라에서 소프트웨어를 유료로 제공하는 업체는 없는 상황이다. 이는 IED의 판매량이 절대적으로 많지 않기에 생기는 현상이면서 동시에 소프트웨어의 기능이 선두 업체의 그것과는 수준의 차이가 있기에 생기는 현상이기도 하다. 소프트웨어의 구현 시 설계도의 개념으로 아키텍처를 먼저 도시해 볼 수 있는데, 이는 소프트웨어 설계자와 사용자간에 Communication 도구로서 사용되어 질 수도 있으며 원하는 품질을 위한 설계도의 역할도 할 수 있다. 이러한 아키텍처를 도시하기 위하여 우선적으로 구현하고자 하는 소프트웨어가 어떠한 기능을 담당해야하는지 빠져서는 안 되는 기능이 있는지 그리고 어떠한 환경의 하드웨어에서 구동되어지는지 등의 기반 사항이 정의 되어야 한다. 여기에는 소프트웨어 구현 중 사용자의 요구에 의해서 기능 정의가 바뀔 것에 대비한 설계까지도 고려 대상이 된다. 지금까지 IED의 기능은 10년 이상의 개발 경험에 의하여 정형화된 틀이 존재하기도 하지만, 계속적인 발전에 의해 새롭게 사용자가 요구하는 기능 또한 계속 추가 되어 왔기 때문에 소프트웨어 또한 그러한 기능을 수용하여 개발해야 된다. 우선 IED의 관리 소프트웨어로서 필요로 하는 기능은 어떠한 것이 있는지 알아본 후, 그러한 기능을 구현하기 위하여 아키텍처는 어떤 방식으로 도시하게 되는지 알아보고자 한다.

2. 본 론

2.1 IED관리 소프트웨어의 아키텍처 원칙

IED관리 도구로서 소프트웨어의 기능으로 대표적으로 적용되어야 하는 것은 차후 개발되는 IED에도 적용이 가능한 통합적 소프트웨어 이어야 한다는 것과 여러 가지 통신 방식을 통하여 실시간으로 설정과 모니터링이 가능하여야 한다는 것이다. 이를 위한 소프트웨어의 일반적인 기본 원칙은 Code readability 극대화, 유지보수성 극대화로 표현되어 질 수 있다. 이는 소프트웨어 내부적으로 재사용성과 유지 보수성을 높이기 위해 추구되는데 일반적으로 이러한 원칙들은 소프트웨어의 성능과 반비례 관계에 있기 때문에 아래에 제시되는 하위 원칙과 design pattern을 이용하여 적절한 품질을 유지 시킬 수 있다.

2.1.1 SRP(Single Responsibility Principle)

OOD(Object Oriented Design)의 기본 원칙으로 각 개체는 단일 역할 또는 책임만을 지도록 하는 원칙이다. 응집도를 높이기 위한 기반이 된다. 이는 클래스나 모듈은 하나의 이유에 의해서만 변경되어야 한다는 원리로 두개 이상의 책임을 하는 경우 변하는 책임에 의해 변하지 않는 책임도 영향을 받아 연쇄적인 수정이 가해질 수 있기 때문에 정해진 원칙이다.

2.1.2 DIP(Dependency Inversion Principle)

모듈의 물리 구조상 논리 구조와는 정반대의 의존성을 취하기 마련인 부분에 일정 기법을 이용하여 이들을 순방향(상위모듈에서 하위모듈로)으로 바꿔준다. 이것은 상위모듈은 하위모듈에 의존해서는 안 되며 상위모듈과 하위모듈 모두 추상화를 통해 의존해야 한다는 것이다. 이는 차후 프로그램의 수정이 발생할 시 수정해야 되는 영역을 한정시키는 효과가 있다.

2.1.3 ISP(Interface Segregation Principle)

인터페이스 분리의 원칙이다. 이는 기능의 분리로 볼 수 있는데, input과 output을 interface라 생각해 본다면 하나의 interface에는 하나의 입력 또는 출력을 할당 하는 것이 시스템 확장 시에 필요한 기능에 대한 변경이나 확장만으로 변화를 최소화 시킬 수 있다는 점과 하나의 방대한 프로그램으로 구성할 때 생길 수 있는 트러블을 예방 할 수 있다는 장점이 있다.

2.2 주요한 Design Patterns

소프트웨어 아키텍처에서는 이미 정형화된 Pattern들이 존재한다. 이는 소프트웨어 개발기술의 발전으로 많은 사람들이 연구하여 이룩해 놓은 것이다. 이렇게 만들어진 Pattern들은 프로그램 사용자의 요구에 따라 적당한 활용을 통해 기능을 효과적으로 구현 할 수 있도록 도와준다. IED용 PC 소프트웨어도 여러 가지 기능을 필요로 하기 때문에 이러한 Pattern을 적용하면 더 효과적으로 개발 할 수 있다.

2.2.1 Strategy Pattern

알고리즘군을 정의하고 각각을 캡슐화하여 교환해서 사용할 수 있도록 만드는 Pattern이다. 이는 자유로운 behavior를 추가할 수 있도록 하는 패턴으로 다양한 통신 프로토콜을 지원해야 하는 IED관리 소프트웨어서 본 Pattern을 이용해서 효과적으로 구현을 할 수 있게 된다.

2.2.2 Façade Pattern

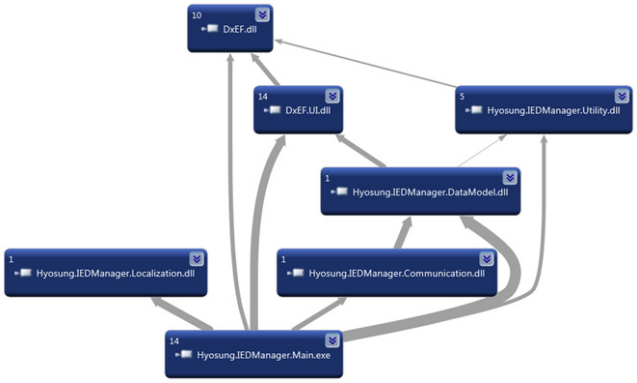
파사드 패턴은 내부의 복잡한 논리를 모르더라도 client에서 해당 기능을 용이하게 사용 할 수 있도록 하는 패턴이다. 이는 서브시스템에 있는 인터페이스 집합에 대해서 하나의 통합된 인터페이스를 제공하는 패턴으로 서브시스템을 좀더 편하게 만드는 상위 수준의 인터페이스를 정의해 준다. IED가 가지는 많은 양의 데이터에서 읽기 또는 쓰기 기능 등을 구현하기 위해 사용한다.

2.2.3 Factory Method Pattern

객체를 생성하기 위해 인터페이스를 정의하지만, 어떤 클래스의 인스턴스를 생성할 지에 대한 결정은 서브클래스가 내리도록 한다. 구체적인 타입을 모르고도 해당 기능을 생성할 수 있도록 하는 패턴이다. 실제 UI(User Interface)에서는 통신 프로토콜이 어떤 것을 이용하는지, 그리고 Serial을 이용한 통신인지 Ethernet을 이용한 통신인지 여부를 모르더라도 이들을 생성하고 사용할 수 있게 한다.

2.3 주요 컴포넌트 및 관계

소프트웨어의 대부분의 컴포넌트 간 관계는 일정한 논리에 따라 구성되는 것이 좋다. 그리고 참조 관계는 단방향을 이루는 것, 즉 상호 참조 또는 순환 참조는 없는 것이 바람직하다. 본 소프트웨어의 구현에서는 위에 설명한 아키텍처 원칙에 따라 컴포넌트를 구성하게 되어 필연적으로 일정한 논리 및 단방향성의 관계를 가지게 된다. 실제 구현의 예는 아래의 <그림 1>과 같이 표현 될 수 있다.



<그림 1> 컴포넌트 및 관계

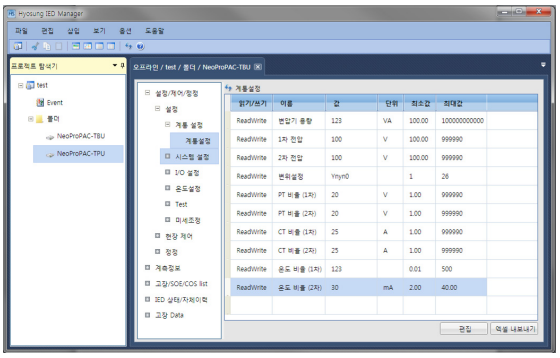
상위에 위치한 컴포넌트일수록 타 컴포넌트에 대한 의존성이 떨어짐과 동시에 하위로 갈수록 타 컴포넌트에 대한 의존성이 강화되는 모습이다.

2.4 소프트웨어의 구현

위에서 설명한 소프트웨어의 구현 원칙과 더불어 디자인 패턴들을 이용하여 구현한 소프트웨어는 아래의 예와 같이 나타날 수 있다.

2.4.1 UI(User Interface)의 구현

사용자가 소프트웨어를 사용하기 위하여 직면하는 UI는 사용자 위주의 설계로 직관적으로 사용하기 편해야 한다. 그러면서 동시에 소프트웨어 설계 기준에 따라 여러 사항을 고려해야 한다. 기본적으로 소프트웨어의 성능이란 것은 화려한 장식 또는 표현과는 반비례하기 때문에 적절한 선에서 절제 되어야 한다.

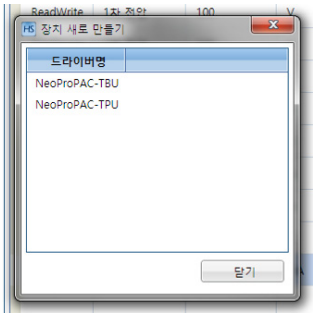


<그림 2> 소프트웨어 기본 UI

그렇기 때문에 위 <그림 2>에서 보듯이 실제 소프트웨어 구현에 있어서 Data를 그래픽이나 그림으로 표현하기 보다는 List형식으로 나타내어 시스템의 리소스를 최대한 적게 사용하는 것으로 하였다.

2.4.2 IED 추가 개발 고려

소프트웨어를 개발 시 아주 중요하게 생각해야 될 사항 중 하나가 차후 update를 고려해야 한다는 것이다. IED의 추가적인 개발 시 Driver 형태로 파일을 추가하면 바로 사용이 가능한 구성을 가지기 위해 Design Pattern을 사용하였다.



<그림 3> IED의 추가

위 <그림 3>의 IED 추가 창과 같이 차후 개발 되는 IED는 Driver를 추가하여 소프트웨어의 재사용성을 높일 수 있다. 이러한 Driver는 손쉬운 제작을 위하여 xml형태로 작성 후 자료 은닉을 위해 변환을 거쳐 사용하게 된다. xml형태로 만든 Driver는 비단 IED뿐 아니라 통신상의 Data를 추가 하거나 통신 방식을 추가하는데 같은 방식으로 사용할 수 있다.

3. 결 론

IED를 개발함에 있어 PC용 관리 소프트웨어의 개발은 필수적인 사항이다. 이때 개발 방향을 어떻게 잡느냐에 따라 차후 작업량이 달라질 수 있음을 알았다. 이는 개발 프로젝트에 있어 개발 기간과 비용을 줄이는데 도움을 줄 수 있음을 뜻하면서 같은 자원으로 더 나은 결과물을 얻을 수 있다는 뜻이 되기도 한다.

본 논문에서는 PC의 소프트웨어 개발에 있어서 설계 단계의 정의에 대해 아키텍처를 논하였지만, 이를 IED전체의 Embedded System에도 적용하면 유사한 효과를 얻을 수 있으리라 판단된다.

[참 고 문 헌]

[1] 렌 베스 외, “소프트웨어 아키텍처”, 에이콘, 2007년
[2] Paul Clements 외, “Evaluating Software Archietcture”, Addison-Wesley, 2005
[3] 김성식 외, “효성 IED를 위한 통합형 파형 분석 툴 개발”, 대한전기학회, 제37회 학술대회 논문집A, 462~463, 2006.7