확장 블록 기반의 장비 프로세스 제어 연구

심민석° 유대승 박성규 김종환 이명재
울산대학교 컴퓨터정보통신공학부
{sms, icoddy, yds, bearknight, ymj}@mail.ulsan.ac.kr

The Control of Instrument's Process based on eXtension Block Markup Language (XBML)

Minsuck Sim° Daesung Yoo Sunghue Park Jonghwan Kim Myeongjae Yi
School of Computer Engineering & Information Technology, University of Ulsan

요 약

본 논문은 확장 블록(eXtension Block) 개념을 사용하여 분산 장비들에 대한 중앙 집중적인 제어 모델을 제시한다. 이런 확장 블록은 OPC와 XML 기술을 사용하여 상위 레벨에서의 추상화를 기법을 사용하여 모델링 함으로써, OPC의 기반 기술인 COM(Common Object Model) 기술과 언어 중립적인 XML 기술이 가지는 이점을 충분히 활용할 수 있는 구조로 디자인 되었다.

1. 서 론

현재 제조 시스템은 다양한 소비자의 요구들을 만족시키고, 생산성 향상을 위해서 점점 더 효율적인 구조로 진화하고 있다[1]. 이러한 구조는 다양하고 복잡한 장비들을 설비(여러 장비를 조합하고 배치하여 시스템을 구성하는 부분)하고 제어(구성된 장비를 제어를 하는 부분)하여 소기의 목적을 달성한다. 그러나 시간이 흐를수록 장비들의 설비와 제어는 점점 더 복잡하고 어려워지고 있다. 이런 상황에서 시스템을 설비하는 부분은 장비들의 규격화, 일반화를 통하여 복잡함을 해결하고 있으나, 제어 부분은 장비 종속적인 제어 기술과 제어 표준의 미비로 인하여 마땅한 해결 방안을 갖지 못한 실정이다.

최근에 일반적으로 사용하는 장비 제어 기술은 장비에 대한 프로토콜 분석을 통해서 얻은 제어 구조와 정보를 사용하여 장비를 제어하는 방법이 주류를 이룬다. 이런 제어 시스템들은 점차적으로 서로 다른 프로토콜 기반의 제어 구조형식을 가지는 다양한 셀들로 구성되는 경향이 있는데, 이런 이유로 인하여 제어 시스템을 개발하고 유지보수 하는데 더욱 더 많은 비용을 사용하고 있는 실정이다. 더욱이 여러 셀들로 구성된 복잡한 시스템에 대해서 시스템 전반에 대한 제어 로직을 쉽게 적용하고 변경하는 것은 거의 불가능한 일이다. 이러한 상황에서 많은 연구가와 기업들이 복잡한 제어기술을 단순화 및 표준화로의 전환에 초점을 두었고, 이러한 결과, 추상화 기법(Abstraction Technology)을 사용하는 방법이 가장 강력하게 대두되고 있다. 장비제어에서 추상화 관련 연구들을 살펴보면 장비의 행위를 하나의 블록으로 정의하고 장비 제어에 대한 기본적인 단위로 사용한다. 이러한 블록은 제어루틴 사용으로 중첩 및 재사용 가능한 구조를 가지며 이러한 블록의 특성으로 인하여 제어 시스템은 다양한 블록의 조합으로 이루어 진다. 이러한 추상적인 개념을 사용한 블록의 특성으로 인하여 복잡해지는 제어 시스템을 쉽고 편리하게 생성하고 유지보수 할 수 있다. 그러나 이런 연구에서 사용된 블록은 프로토콜 레벨의 추상화와 각기 다른 블록 형식으로 인하여 블록 구성이 힘들 뿐만 아니라 서로 다른 벤더에서 생성한 블록들의 조합에 많은 어려움이 있는 상황이다. 따라서 현재 제조 시스템들은 사용자의 다양한 요구사항과 쉽고 편리하게 제어 로직을 표현하고 적용할 수 있는 개방 구조(Open-Architecture) 기반 제어 시스템으로의 전환을 기도하고 있다[2].

본 논문은 상위레벨의 추상화를 통하여 쉽고 편리하게 장비를 제어하는 확장 블록(eXtension Block) 기반의 제어 소프트웨어 개발 방법을 제시하고, 이런 방법론을 사용하면 쉽게 개방구조(Open-Architecture)의 제어 시스템으로 전환할 수 있다는 것을 증명한다. 본 논문에서 제시하는 개방구조(Open-Architecture)는 전반적인 시스템에 대한 제어 로직을 처리하기 위해 중앙 집중적인 제어 방식을 취하고 있으며, 장비 제어 표준으로 나타나는 OPC(OLE for process control)와 장비의 특성 및 행위를 언어 중립적으로 기술할 수 있는 XML (eXten-sion Markup Language)을 사용하여 장비와 시스템들간의 수직 및 수평 통합을 할 수 있는 구조를 가진다. 또한 제어 시스템을 구성하는 확장 블록에 대한 재사용성(Reusability)을 높이고, 레거시 시스템들과 쉽게 통합 하며, 일반 소프트웨어 개발 자들이 쉽게 제어 소프트웨어를 만들 수 있는 친밀성을 높이는데 초점을 둔다.

2. 관련 연구

XML을 사용하여 추상화한 연구 중 대표적인 연구(IML&AIML, CoML, DDL, CMIML)를 살펴 보고, 본 논문에서 제안하는 확장 블록과 비교하면 다음과 같다.

NASA GSFC(Goddard Space Flight Center)[3]는 플랫폼 독립적인 XML 기술을 사용하여 장비의 특성, 제어 명령, 메시지 형태, 통신 규칙을 포함하는 IML(Instrument Markup Language)을 정의하였고, 실제로 원격(남극, 고산 지대)에 존재하는 천문학 장비에 적용시킨 AIML(Astronomical Instrument Markup Language)을 생성하기도 하였다. 특히 AIML는 GUI 생성 정보를 포함함으로써 제어 소프트웨어의 생성을 지원하는 특성을 가진다. 이외에 NASA에서는 IML을 사용하여 원격에 존재하는 장비들을 제어하는 프레임웍 생성에 관여하는 ICML(Instrument Control Markup Language)등

본 연구는 한국과학재단 지정 울산대학교 네트워크 기반 자동화연구센터의 지원에 의해 이루어졌습니다.

의 연구도진행 중이다.

CoML(CANopen Markup Language)[4]은 필드 버스 프로토콜 중의 하나인 CAN 인터페이스(CANSetup, StateInformation, Module)를 XML을 이용하여 기술하고, CoML문서를 처리 할 수 있는 CANINSIGHT 시스템을 사용하여 장비를 제어하였다. CANSetup는 EDS문서에 포함된 디바이스(Device)의 데이터 형태, 제어 형태, 기본 값 등의 정보를 가지고 있으며, 특별히 EDS2CoML 변환기를 사용하여 정보 변경 가능하다. StateInformation는 데이터의 처리, 인자의 구조 및 값을 가지며, Module은 개별적인 디바이스의 정보를 저장하는 부분이다.

DDL(Device Definition Language)[5]은 필드 버스를 사용하는 장비들에 대해서 동일한 제어방식을 사용하여 제어 하도록 정의한 부분이며, 필드 버스의 대표적인 HART Communication Foundation, the Fieldbus Foundation과 Profibus International에서 제안하였다. DDL은 장비를 기술하는 부분(Device Description)과 기술한 DDL문서를 읽어서 처리하는 DDS(Device Description Services) 부분으로 분류한다. DDL 문서의 내용은 3가지 타입의 블록(Resource block, function block, transducer block)으로 구성한다. Resource block은 필드버스 장비의 특성을 가지는 부분(장비의 이름, 제조회사, 시리얼 넘버)등을 포함하며 하나의 장비에 하나의 Resource block을 가진다. Function block은 제어 시스템에게 장비의 행위를 기술한 블록이며 사용자에 의해 스케줄링 된 FFB(Flexible Function Block)을 만들고 재사용할 수 있다. Transducer block은 장비의 구성정보(Sensor type)를 포함하는 부분이다. Foundation Fieldbus에서는 추가적으로 Capability File을 제공하여 장비(device)의 변수의 특징을 볼 수 있는 부분을 제공한다. 이렇게 DDL를 기반으로 추가적인 특성들이 존재하는데 이러한 것들을 포함하고 공통적으로 나가기 위하여 EDDL(Electronic Device Description Language)을 구성하는 연구를 하고 있다.

CMIML (Control & Monitoring Instrument Markup Language)[6]은 API 레벨에서 생산현장의 다양한 장비와 설비들의 제어 및 모니터링 소프트웨어에 대한 쉬운 개발과 유지보수성을 향상을 위해서 장비의 제어정보 및 사용자 인터페이스 정보, 모니터링 정보, 장비와 PC사이의 통신방법, 장비의 스케줄링 정보를 XML 기술을 사용하여 모델링 하였으며 이를 처리 하기 위하여 CMIML 프레임웍을 제공하였다.

본 논문에서 제안하는 확장 블록은 OPC 컴포넌트의 장비의 데이터 엑세스 구조를 가지는 데이터를 여러 개의 블록으로 포장하였다. 이런 반면 CoML과 DDL은 프로토콜 레벨에서 블록으로 모델링 하였으며, IML&AIML과 CMIML등은 API 레벨의 모델링에 초점이 맞춰져 있다.

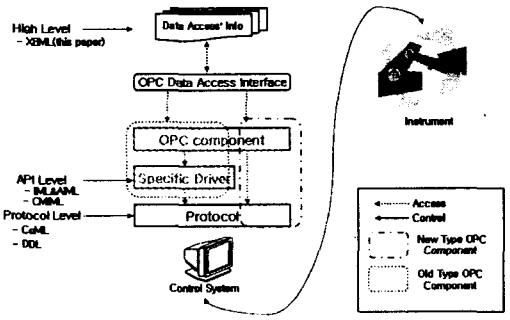

그림 1. 블록 생성에 관한 기존 연구와 비교 구문

3. 확장 블록 기반의 장비 제어 모델

본 논문에서 제안하는 확장 블록(eXtension Block)은 장비 제어의 기본 단위이며 여러 블록들을 스케줄링 할 수 있는 중첩된 구조와 재사용 하기 편리하게 설계 되었다. 확장 블록은 기본타입과 확장 타입의 형태로 분류할 수 있다.

기본 타입은 장비가 가질 수 있는 하나의 요소(제어 정보, 예를 들면 OPC-Server-Name.Group.Item의 형식)의 정보와 선택한 요소가 가질 수 있는 값의 집합을 중심으로 여러 정보(Block Information, View, Basic OPC Component,)가 결합된 형태이다. 이렇게 구성한 제어 정보와 값들로 인하여 장비는 행위를 하게 된다.

확장 타입은 다양한 블록들-기능 블록(Function Block), 컨트롤 블록(Control Block), 진단 블록(Diagnosis Block)-을 사용하여 또 다른 하나의 확장 블록으로 재구성된 블록이다. 재구성 정보는 장비 제어 로직(Instrument Control Logic)으로 표현되며 이 정보를 중심으로 여러 정보(Block Information, View, Basic OPC Component, Reference Block,)가 결합된 형태를 가진다.

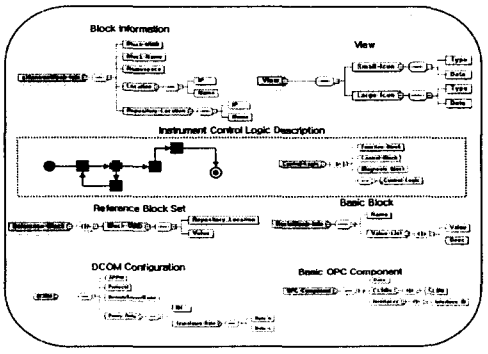

그림 2. 확장 블록 구조

확장 블록 디자인 시 특별한 뷰(View)와 기본 OPC 컴포넌트(Basic OPC Component) 부분에 기입하는 정보는 기존에 많이 사용하는 형식(파일의 위치 정보)이 아니라 파일의 내용(Image Data) 자체를 태그 사이에 기입하는 형태로 디자인 하였다. 물론 이러한 시도가 많은 단점을 가지고 있지만 인터넷(80포트)을 사용하여 방화벽을 자유롭게 통과하여 어디서든지 이동할 수 있고, 확장 블록을 지원하는 툴(iMaker)에서 이미지나 컴포넌트를 재구성하여 사용할 수 있는 이점를 가지고 있어서 본 논문에서는 이러한 방법을 채택하였다.

확장 블록의 정보는 각기 분산되어 있는 저장소(eXtension Block's Repository)에 저장되도록 설계 하였다. 저장소는 XML 형태의 저장 카테고리와 이를 관리 하는 확장 블록 저장 컴포넌트(eXtension Block Storage Component)로 구성한다. 확장 블록 저장 컴포넌트는 윈도우 서비스 형태로 디자인되었다. 아래의 그림 [그림 3]는 확장 블록(eXtension Block)이 자신을 구성하는 다른 확장 블록들의 정보(다른 저장소에 저장되어 있는 블록 정보)를 가져오는 과정을 기술한 것이다.

확장 블록(eXtension Block)은 OPC 래퍼 컴포넌트(OPC Wrapper Component)에 의해서 해석되며, 확장 블록을 구성하는 다른 확장 블록이 있는 지 없는 지 알아본다. 만약 있으면 구성되는 블록의 식별자(UUID) 값과 저장되어 있는 저장소(Repository)의 위치 정보를 취득한다(0). 현재 참조하는 저장소를 관리 하는 컴포넌트(eXtension Block Storage Component)를 이용하여 참조하는 블록

정보를 취득한다(1, 1.1.x). 현재 참조하지 않는 저장소에 있는 확장 블록은 컴포넌트의 위치 투명성을 사용하여 원격에 있는 확장 블록을 관리하는 컴포넌터에 접속하고(1.2) 결과를 OPC 래퍼 컴포넌트가 참조할 수 있는 곳으로 읽어 와서(1.2.x) 확장 블록을 구성하는 모든 구성 요소를 갖추게 된다.(2)

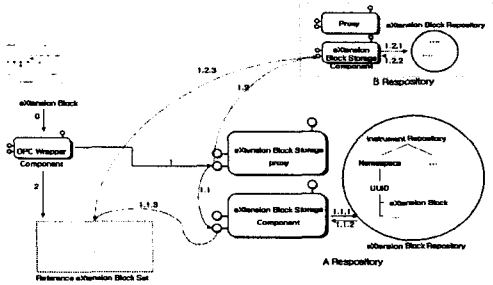

그림 3. 저장소에서 확장 블록을 참조하는 과정

이런 과정을 통하여 OPC 레퍼 컴포넌트는 블록의 정보들을 분석하여 OPC 컴포넌트를 재구성하고 레지스트리에 정보를 추가한다. 이때 재구성한 OPC 컴포넌트는 확장 블록에 명시된 컴퓨터의 IP(Computer's IP)에 존재하는 OPC 컴포넌트와 연결할 수 있는 OPC 컴포넌트의 플락시 역활을 한다. (COM은 COM 컴포넌트에접근할 수 있는 플록시 역할을 하는 부분을 포함) 이러한 과정을 거치면서 하나의 블록에 의존적인 여러 블록에 연결되어 있는 컴포넌트들과의 연동을 할 수 있다.

[그림 4]는 확장 블록을 지원툴(iMaker)을 사용하여 생성한 확장 블록이 DCOM 프로토콜을 사용하여 원격에 존재하는 장비를 제어하는 과정을 기술한 것이다.

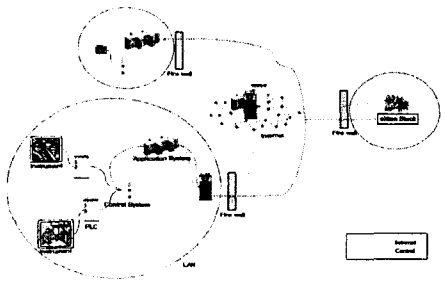

그림 4. 확장 블록을 사용한 장비 제어

우리가 연구하는 제어 시스템의 모델은 원격에 존재하는 작은 자동화 시스템(Automatic system)들로 이루어 져있고, 각각의 자동화 시스템은 서로 다른 셀들로 구성된다. 셀들로 구성된 장비들은 OPC 기반의 장비 제어를 따르고 있으며 각각의 셀들은 OPC 서버를 통해서 제어 된다. 또한 이런 자동화 시스템은 분산된 지역에 다수의 OPC 서버를 통하여 제어 하는 [그림 5]와 같은시나리오를 생각할 수 있다.

4. 결론 및 향후 연구 과제

우리는 Shim[7]이 제시한 블록 개념을 확장하여, 분산 환경에서 복잡하게 서로 묶여 있는 장비들에 대하여 쉽고 편리하게 제어 시스템을 생성하는 방법을 정의 하였으며, 제안 방법(eXtension Block)의처리를 위하여 프레임웍(*i*Maker)을 제시 하였다. *i*Maker 시스템

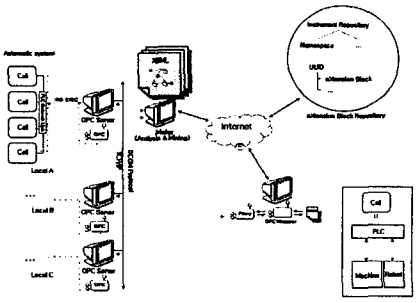

그림 5. 장비 제어 시나리오

은 OPC 서버 컴포넌트에서 장비의 특성 정보 추출하고 XML 기술을 사용하여 장비의 행동정보를 표현 및 재구성하여 장비 제어 소프트웨어 질을 향상시킬 수 있고 유지보수에 대한 비용을 현저히 줄일 수 있다. 확장 블록 기반의 제어 소프트웨어 개발 방법은 아래와 같은 이점을 가질 수 있다.

- 개발 & 유지 보수 비용 감소
- 재사용성 증가
- 전문적인 하드웨어 지식이 없어도 제어 소프트웨어 개발 가능
- 시뮬레이션 기능의 사용으로 인하여 장비 없이도 장비 제어 로직 개발 가능
- 장비 제어 분야에 대한 거리감 해소

본 논문에서 제안하는 방법은 생산 기술의 변화와 컴퓨터 관련 기술의 급격한 발달에 빠르고 적응하고 유연성(flexibility), 통합성(Integration) 및 동시성(concurrency)를 만족시키는 개방구조(Open architecture)로의 전환에있어서 중요한 기술이 될 것이라 생각된다.

향후 연구 과제로는 ***i*Maker** 시스템을 중앙 집중적인 제어 구조에서 분산 구조로의 변환 및 배치 (Deployment)에 초점을 둔다.

본 연구는 한국과학재단 지정 울산대학교 네트워크 기반 자동화연구센터의 지원에 의해 이루어졌습니다.

[참고문헌]

[1] W.Sperling and P.Lulz, "Enabling open control systems: An introduction to the OSACA system platform", ESPRIT III Project: Stuttgart: FISW GmbH, 1995

[2] P.K. Wright, "Principles of open-architecture manufacturing", Journal of Manufacturing System, vol. 14, no.3, pp. 187-202, 1995

[3] NASA, Astronomical Instrument Markup Language, "pioneer.gsfc.nasa.gov/public/aiml/, iml/"

[4] Buhler, D., "The CANopen Markup Language representing fieldbus data with XML", Industrial Electronics Society, 2000. IECON 2000. 26th Annual Confjerence of the IEEE , Volume: 4 , 2000

[5] DDL(Device Description Language), ProfiBus, "www.profibus.com"

[6] Dae-Sung yoo, Min-Suck Sim, Sung-ghue Park, Jong-Hwan Kim, Myeong-Jae Yi, "A Framework for automatic generation of instrument control and monitoring software", Proceedings of the 7th Korea-Russia International Symposium, KORUS 2003, vol2. pp, 428-432

[7] Minsuck Shim, Sungkyu Park, Dae-Sung Yoo, Jong-Hwan Kim, Myeongjae Yi, " A Study on the Flexible and Efficient Instrument Control Software Generation ", Proceedings of the 7th Korea-Russia International Symposium, KORUS 2003, vol2. pp, 447-452