제어 소프트웨어의 생성을 위한 Generator의 설계 및 구현

유대승*, 심민석*, 박성규*, 김종환*, 이명재*
*울산대학교 정보통신공학과
e-mail:ooseyds@mail.ulsan.ac.kr

The Design and Implementation of Generator for Generating Control Software

Dea-Sung Yoo*, Min-Suck Sim*, Sung-Ghue Park*, Jong-Hwan Kim*, Myoung-Jae Yi*
*School of Computer Engineering & Information Technology, University of Ulsan

요 약

과거 우리는 플랫폼 독립적인 XML과 GUI 기반의 툴들을 제공하여 장비에 대한 제어 및 모니터링 소프트웨어의 효율적인 생성과 유지보수성을 향상하기 위한 프레임워크[1][2]를 제안하였다. 제안한 프레임워크는 세 가지의 XML 문서(IID, MAP, CMIML), VI Wizard, Generator로 구성되었다. 본 논문에서는 제안한 프레임워크의 한 구성요소인 Generator에 대한 설계와 구현을 보인다. Generator는 장비의 제어정보, 모니터링 정보, 통신 정보, 스케줄 정보, 제어 소프트웨어의 사용자 인터페이스 정보 등을 기술하는 CMIML(Control & Monitoring Instrument Markup Language) 문서를 이용해서 GUI 기반의 사용자 인터페이스 편집 환경을 제공하고, CMIML 문서를 소프트웨어 코드로 변환함으로써 제어 및 모니터링 소프트웨어를 자동 생성한다.

1. 서 론

산업의 발전과 그에 따른 작업 환경의 변화는 점점 생산 현장에서 사용되는 장비들을 자동화시켜갔고 이런 자동화 장비들은 필드버스, Ethernet과 같은 산업용 네트워크를 통해서 서로 연결이 되어 크고 복잡한 설비를 이루게 된다. 다양하고 복잡해져가는 설비에 대한 문제를 해결하기 위해서 장비들은 규격화되고 일반화되어 가고 있지만 장비 종속적인 제어기술과 제어방법에 대한 표준은 아직 미비한 상황이며 복잡한 설비들을 제어하기 위하여 장비들을 직접 제어하기보다는 PC 기반으로 제어하게 됨에 따라 이런 자동화 장비들의 제어 및 모니터링을 위한 소프트웨어의 중요성이 점차 증대되고 있다.

다양한 네트워크를 통해서 연결된 많은 자동화 장비들의 제어와 모니터링을 위한 소프트웨어의 개발에는 몇 가지 문제가 존재한다. 자동화 장비들은 제조사 별로 제공하는 드라이버 API가 다르기 때문에 복잡한 설비를 이루고 있는 복합 장비들을 통합적으로 제어하기 위하여 장비별로 별도의 소프트웨어를 개발해야 한다. 또한 장비의 드라이버 API가 변경되거나 새로운 요구가 발생되는 경우 재개발해야 한다는 어려움이 있다. 그리고 장비들을 연결하는 네트워크와 장비가 지원하는 플랫폼이 다양하기 때문에 소프트웨어가 다양한 네트워크와 플랫폼을 지원하도록 개발되어야 하는 문제가 있다. 이런 문제들로 인해서 자동화 장비들의 제어와 모니터링을 위한 소프트웨어의 개발과 유지보수에 많은 비용이 소요된다.

이에 우리는 이런 문제들을 해결하기 위해서 효율적인 제어 및 모니터링 소프트웨어의 개발을 지원하는 프레임웍을 제안하였다[1][2]. 제안한 프레임워크은 세 가지의 XML 문서(IID, MAP, CMIML)와 두 가지의 도구들(VI Wizard, Generator)로 구성되었다. VI Wizard는 장비에 대한 인터페이스 정보를 기술한 IID를 이용하여 GUI기반으로 VI를 구성하고, 새롭게 구성된 VI를 CMIML로 변환하는

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

도구이다. VI Wizard를 통하여 생성된 CMIML은 Code Generator에 의해서 특정한 플랫폼에 맞는 소스코드로 변환되어 운용 될 것이다.

본 논문에서는 우리가 제안한 프레임워크의 구성요소인 Generator에 대한 설계와 구현을 소개한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구와 연구동향에 대해서 살펴보고, 3장에서는 Generator에 대하여 세부적으로 설명한다. 4장에서는 제어 및 모니터링 소프트웨어의 생성 과정을 보이고, 5장에서는 결론 및 향후 연구 과제에 대해서 살펴보도록 한다.

2. 관련 연구

기존의 산업 현장에서 사용되던 장비나 설비는 내부에 제어를 위한 프로세서를 탑재하고 있었고 사용자가 직접 조작을 하여 장비나 설비를 가동시켰다. 이러한 기존의 장비나 설비는 상당히 고가였고, 유지보수에도 상당한 비용을 소모하게 되었다. 따라서 산업체들은 비용의 절감을 위해 장비나 설비의 자동화에 관심을 갖게 되었고, 자동화 장비의 구현이나 장비들을 연결하는 프로토콜의 구현에 대한 연구가 진행되어 왔다. 90년대 초 PC기반의 자동제어 시스템[3]에 대한 연구가 시작되었고, 90년대 후반, 2000년에 이르러 본격적으로 PC기반의 제어 시스템의 개발에 대한 연구[4][5][6]가 이루어지기 시작했다. 그러나 산업현장에서 사용되어지는 장비들의 자동화에 대한 관심이 커지고 많은 장비들이 자동화되어져 감에도 불구하고 국내에서는 이런 자동화 장비들의 제어와 모니터링을 위한 소프트웨어의 개발에 대한 연구가 거의 전무한 상황이다.

국내에 비해서 국외에서는 이미 많은 연구가 진행되어 오고 있고, 이미 상용화 되어있는 관련 소프트웨어들이 출시되어 있다. 최근에는 NASA에서 천체 관측에 사용되어지는 장비들의 제어를 위한 새로운 Markup언어를 제안하는 등의 장비나 설비의 인터페이스 또는 산업용 네트워크의 프로토콜인 필드버스를 XML로 표현하는 방법들이 제안되고 있다.

상용화되어 있는 소프트웨어 중 대표적인 것에는 NI(National Instrument)사의 LabVIEW[7]와 Rockwell소프트의 RsVIEW[8]가 있다. 이 두 소프트웨어는 기본적으로 자동화 장비의 제어와 모니터링을 위한 소프트웨어를 생성하는 기능을 수행하며, 개발자가 직접 소프트웨어의 코드를 작성하는 것이 아니라 GUI 기반의 VI(Virtual Instrument) 편집 환경을 제공함으로써 쉽고 빠른 개발과 유지보수를 지원하고 있다.

그러나 이런 소프트웨어들은 장비 제조업체에서 제공하는 API를 사용하여야 하고 장비의 API가 변경되면 변경된 API를 다시 제공받아야 하는 문제가 있다. 그래서 장비의 API를 XML 문서로 기술함으로써 이런 문제들을 해결하고자 하는 노력이 있었다. 대표적으로 NASA의 IML(Instrument Markup Language)[9]과 AIML(Astronomical Instrument Markup Language)[10]이 있다. 천체 관측에 사용하는 장비들의 인터페이스를 XML 문서로 기술하고, 이 정보와 API의 정보를 연결시킴으로써 API의 변경이 제어 소프트웨어에 끼치는 영향을 최소화시키고 쉽고 빠른 개발을 지원하도록 한다.

근래에는 산업용 네트워크의 프로토콜 중 가장 각광을 받고 있는 필드버스에 대한 연구가 가장 활발하게 이루어지고 있다. 대표적인 것으로 필드버스의 표준 프로토콜 중 하나인 CAN(Controller Area Network)에 대한 XML 어플리케이션인 CoML(CANopen Markup Language)[11]이 있고, 이 CoML을 이용해서 구현한 CANINSIGHT 시스템[12]이 있다. 그러나 필드버스는 독립적으로 구성된 서로 다른 시스템들 사이에 유연성이나 호환성이 많이 결여되어 있다. 그래서 이런 문제들을 해결하기 위한 연구들이 진행되고 있으며, 대표적으로 Tag-based Trees를 이용해서 서로 다른 필드버스들을 상호 연동시키는 방법에 대한 연구[13]와 필드버스 내의 장비와 시스템 사이의 통신 인터페이스의 표준화에 대한 연구[14]가 진행 중에 있다.

위와 같은 국내외 연구에서 보듯이 자동화 장비의 드라이버 API에 의존적이지 않으면서, GUI 환경에서 쉽고 빠르게 제어 소프트웨어를 개발할 수 있도록 하는 환경이 필요하다.

3. Generator

Generator는 제안한 프레임워크의 VI Wizard에 의해서 생성된 CMIML 문서를 이용해서 GUI 기반의 사용자 인터페이스 편집 환경을 제공하고, CMIML을 소프트웨어 코드로 자동 생성하는 도구이다. Generator를 통해서 장비에 대한 전문적인 지식이 없이도 제어 소프트웨어를 생성할 수 있게 된다.

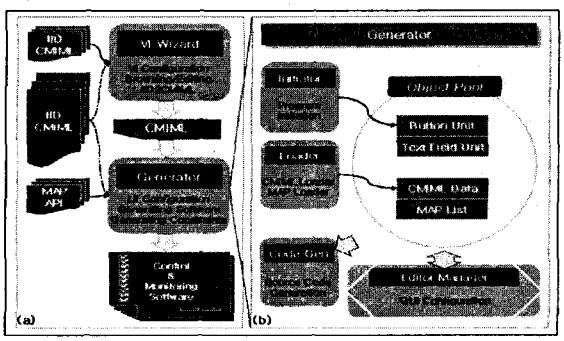

그림 1. Generator 시스템 구성도

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

3.1 Generator System Architecture

그림1의 (a)는 [1][2]에서 제안한 전체 프레임워크를 보이고 있고, (b)는 Generator의 세부적인 시스템 구조도를 보이고 있다.

그림 1의 (b)에서 보는 바와 같이 Generator는 3개의 모듈, 1개의 데이터 저장소, 1개의 매니저로 구성된다.

Generator 시스템의 구성 요소들의 기능을 요약하면 다음과 같다.

· Initiator Module

Generator의 실행에 필요한 초기화 작업을 수행하는 모듈이다. 이 모듈은 처음 실행될 때 실행될 뿐만 아니라 새로운 CMIML을 로드할 때마다 실행이 된다. 새로운 CMIML이 로드될 때 이전에 작업하던 데이터들은 초기화되어져야 하기 때문이다. 주요 수행 기능은 내부 데이터를 초기화하고 GUI 편집 환경에서 사용되어질 Button Unit과 TextField Unit을 Object Pool에 등록하는 것이다.

· Loader Module

Loader Module은 CMIML Loader와 MAP Loader로 구성된다. Generator에서는 하나의 CMIML 문서와 다수의 MAP 문서를 로드하게 된다. CMIML Loader는 CMIML 문서를 읽어서 제어 및 모니터링 소프트웨어 생성에 필요한 요소들을 추출한다. 추출된 요소들은 Object Pool의 CMIML Data에 저장된다. MAP Loader는 MAP 문서를 읽어서 CMIML에 기술된 정보와 실제 API와의 연결정보를 추출한다. 추출된 정보는 Object Pool의 MAP List에 저장된다.

· CodeGen Module

Object Pool에 저장된 데이터들을 사용하여 특정한 플랫폼에 맞는 제어 소프트웨어 코드를 생성하는 모듈이다. 현재 CodeGen 모듈은 자바 코드를 생성하도록 구현되었지만 다양한 CodeGen 모듈의 개발에 의해서 다양한 플랫폼에 대한 코드를 생성할 수 있을 것이다.

· Editor Manager

사용자 인터페이스를 편집할 수 있도록 하는 GUI 편집 환경을 제공한다. Object Pool에 새로운 사용자 인터페이스 객체를 생성하거나, 삭제하는 기능을 수행한다. 이때 생성되는 객체는 Object Pool의 CMIML Data를 참조해서 생성된다.

· Object Pool

Initiator Module과 Loader Module에 의해 추가된 객체와 데이터들을 저장하는 저장소이다. 저장된 객체와 데이터는 Editor Manager에 의해서 사용자 인터페이스 편집에 사용되어지며, 최종적으로 CodeGen Module이 제어 소프트웨어 코드를 생성할 때 사용되어 진다.

3.2 Generator System Implementation

Generator는 Visual Basic 6.0 환경에서 XML 파서로 MSXML3.0을 사용하여 구현하였다.

그림 2는 Generator의 실행 화면이다. 그림 2의 (a)는 Loader Module(CMIML Loader, MAP Loader)에 의해서 읽혀진 CMIML 문서와 MAP 문서의 데이터를 기반으로 생성해낸 제어 소프트웨어의 소스코드를 보여준다. 현재 Generator는 JAVA 기반의 소스코드를 생성하도록 구현

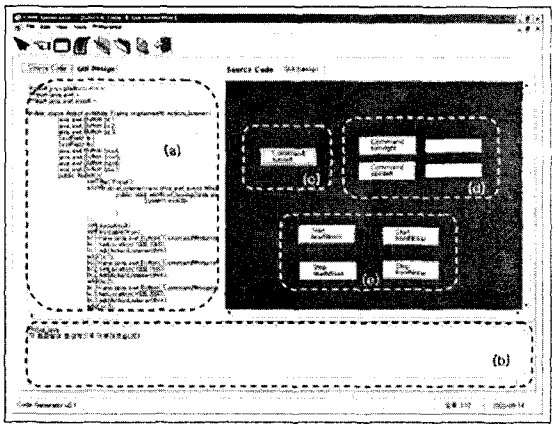

그림 2. Generator의 실행 화면

되었다. (b)는 생성된 소스코드를 외부 컴파일러를 통해 컴파일 했을 때의 결과를 보여주는 화면이다. (c)와 (d)는 일반적인 Command가 사용자 인터페이스를 가질 때를 보여준다. (c)는 Command의 Parameter가 없거나 사용자 인터페이스를 가지지 않는 경우이고, (d)는 Command의 Parameter가 사용자 인터페이스를 가질 때를 보여준다. (e)는 일련의 Command들의 모임인 Schedule이 사용자 인터페이스를 가질 때를 보여준다. Schedule은 여러 동작들의 순차적인 실행이므로 실행 중 중단할 수도 있어야 한다. 따라서 Schedule을 중단할 수 있는 인터페이스도 존재해야함으로, 하나의 Schedule에 대해서 2개의 인터페이스가 생성된다.

4. 제어 및 모니터링 소프트웨어 생성 과정

그림3은 본 논문에서 제안하는 프레임워크를 사용하여 제어 소프트웨어를 생성하는 과정을 보이고 있다. 점선 사각형 부분은 VI Wizard에 의해서 CMIML을 생성하는 과정이고, 실선 사각형 부분은 본 논문에서 소개한 Generator에 의해서 수행되는 부분이다. 가장 먼저 해당 장비의 개발자가 장비의 인터페이스를 분석해서 IID 문서와 MAP 문서를 작성해야 한다. VI Wizard는 IID 문서를 이용하여 GUI 기반의 편집 환경에서 새로

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

운 VI를 구성하고, 구성된 VI를 CMIML 문서로 변환한다. 생성된 CMIML 문서는 다른 VI의 구성을 위해 재사용 될 수도 있다. Generator는 VI Wizard에 의해서 생성된 CMIML 문서를 이용해서 사용자 인터페이스를 편집하고, 제어 소프트웨어 코드를 생성한다.

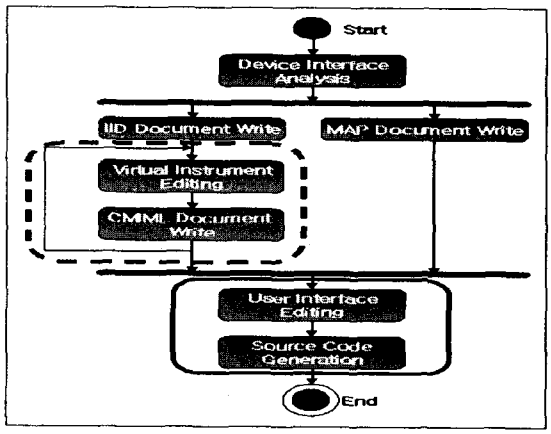

그림 3. 제어 소프트웨어 생성 과정

5. 결론 및 향후 연구과제

본 논문에서는 장비의 인터페이스 정보와, 장비와 PC 사이의 통신 정보, 제어 소프트웨어의 사용자 인터페이스 정보, 장비의 스케줄 정보 등을 기술하고 있는 XML 기반의 CMIML 문서를 이용하여 GUI 기반으로 사용자 인터페이스 편집 환경을 제공하고, 제어 소프트웨어의 소스코드를 자동 생성하는 Generator에 대한 설계와 구현을 보였다.

Generator는 VI Wizard와 함께 사용되어 전문적인 지식들(장비 제어, 제어 소프트웨어, XML) 없이도 자동으로 제어 소프트웨어를 생성함으로써 원격으로 제어되는 생산 현장의 장비나 설비들에 적용될 수 있는 제어 소프트웨어를 적은 비용으로 쉽고 빠르게 개발하고 유지보수 할 수 있을 것이다.

향후에는 현재 자바코드를 생성하는 CodeGen 모듈을 다양한 플랫폼에 맞는 소스코드를 생성할 수 있도록 확장하고 다양한 사용자 인터페이스를 기술할 수 있는 방법에 대한 연구가 필요하다. 또한 소프트웨어 코드의 변경을 최소화하면서 무정지 시스템을 가능하게 하는 실시간 참조 제약사항에 대한 정의와 자동 생성된 소프트웨어의 안정성을 검증할 수 있는 가상 시뮬레이션에 대한 연구가 필요하다.

[참고문헌]

[1] 유대승, 심민석, 박성규, 김종환, 이명재, "제어 및 모니터링 소프트웨어의 효율적인 개발을 위한 프레임워크 설계", 정보과학회춘계학술발표대회, 2003

[2] 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, vol. pp, 428-432

[3] 구영재, 이준서, 이인범, 장근수, 'PC를 이용한 자동제어시스템 개발', 한국자동제어학술회의논문집(KACC), 1991

[4] 변승현, 마복렬, '대용량 플랜트 제어를 위한 PC 기반 I/O 인터페이스 시스템 구축에 관한 연구', 한국자동제어학술회의논문집(KACC), 1999

[5] 김정구, 최경현, 홍금식, 'PC에 기반을 둔 개방형 로봇제어시스템:PC-ORC A PC-Based Open Robot Control System:PCORC', 제어.자동화.시스템공학 논문지, 2000

[6] 박남준, 김홍석, 박종구, 'PC기반의 생상시스템을 위한 운용소프트웨어 구조', 제어.자동화.시스템공학 논문지, 2001

[7] National Instrument "www.ni.com/"

[8] Rockwell Automation "www.rockwell.com/"

[9] NASA "pioneer.gsfc.nasa.gov/public/iml/"

[10] NASA "pioneer.gsfc.nasa.gov/public/aiml/"

[11] Dieter Buhler, "The CANOpenn Markup Language Representing Fieldbus Data with XML", *Industril Electronics Society*, 2000. IECON 2000. 26th Annual Conference of the IEEE, Volume: 4, 22-28 Oct. 2000 Page(s): 2449-2454 vol.4

[12] Dieter Buhler, Wolfgang Kuchlin, "Remote Fieldbus System Management with Java and XML", Industrial Electronics, 2000. ISIE 2000. Proceedings of the 2000 IEEE International Symposium on, Volume: 1, 4-8 Dec. 2000 Page(s): 1-6 vol.1

[13] Steffen Deter, 'Fieldbus Device Description using Tag-based Trees', Africon Conference in Africa, 2002. IEEE AFRICON. 6th, Volume: 1, 2-4 Oct. 2002 page(s): 263-268 vol.1

[14] *FDT Joint Interest Group, 'http://www.fdt-jig.org/index2.html'*

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