제어 및 모니터링 소프트웨어 자동 생성을 위한 프레임워크

유대승° 심민석 박성규 김종환 이명재
울산대학교 컴퓨터 · 정보통신공학부
{ooseyds°, sms, icoddy bearknight, ymj}@mail.ulsan.ac.kr

A Framework for Automatical Generation of Instrument Control & Monitoring Software

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

요 약

생산 현장에서 사용되는 자동화 장비들은 다양한 플랫폼과 통신 방법을 사용하여 운용되므로 제어 및 모니터링을 위한 소프트웨어 개발을 위해서는 전문적인 지식이 요구되고, 개발 및 유지보수에 많은 비용이 소요된다. 본 논문에서는 여러 자동화 장비들의 제어 및 모니터링 소프트웨어에 대한 쉬운 개발과 유지보수성을 향상시킬 수 있는 프레임워크를 제안하고자 한다. 본 연구에서는 제어 및 모니터링 소프트웨어에 대한 자동생성을 위하여 세가지(ICD, MAP, CMIML)의 XML 문서를 제안하며, 이를 이용해서 제어 및 모니터링 소프트웨어를 자동 생성함으로써 자동화 장비와 소프트웨어에 대한 전문적인 지식 없이도 제어 및 모니터링 소프트웨어를 개발하는 것이 가능하고, 유연하고 신뢰성 있는 자동화 시스템을 구축할 수 있다.

1. 서 론

산업의 발전과 그에 따른 작업 환경의 변화는 점점 생산 현장에서 사용되는 장비들을 자동화시켜갔다. 이런 자동화 장비들은 네트워크를 통해서 연결이 되어 크고 복잡한 설비를 이루게 된다. 자동화 장비들을 네트워크를 통해 연결을 하게 되면 시스템의 유연성이 향상되고 분산된 장비들간의 정보를 효율적으로 수집하는 것이 가능하게 된다[1].

다양한 네트워크를 통해서 연결된 자동화 장비들의 제어와 모니터링을 위한 기존의 소프트웨어 개발에는 몇 가지 문제점이 존재한다. 첫 번째는 제어 및 계측 관련 데이터의 양이 너무 방대하다는 점이다. 두 번째는 자동화 장비들이 다양한 플랫폼과 통신방법을 사용하고 있다는 점이다. 이것은 각각의 플랫폼과 통신방법에 따라 다른 소프트웨어를 개발해야하는 문제점을 야기 시킨다. 세 번째는 장비들이 다양한 플랫폼과 통신방법을 지원하기 때문에 각각의 플랫폼과 통신방법에 대한 전문적인 지식을 요구한다는 점이다. 네 번째는 장비의 인터페이스가 바뀌게 되면 소프트웨어를 재개발해야한다는 점이다. 이런 문제점들로 인해서 제어 및 모니터링 소프트웨어의 개발과 유지보수에 많은 노력과 비용이 소요된다.

이에 본 논문에서는 이런 문제점들을 해결하기 위해서 생산 현장에서 운용되는 여러 자동화 장비들의 제어 및 모니터링 소프트웨어에 대한 쉬운 개발과 유지보수성을 향상시킬 수 있는 프레임워크를 제안한다. 본 연구에서 제안하는 프레임워크에서는 제어 및 모니터링 소프트웨어에 대한 자동생성을 위하여 세가지(ICD, MAP, CMIML)의 XML 문서를 작성한다. ICD는 장비에 대한 인터페이스 정보를 기술하고, MAP은 ICD에 기술된 인터페이스와 실제 장비 드라이버 API 또는 Address Map의 연결 정보를 기술하며, CMIML은 작성된 ICD파일을 이용해서 제어정보, 사용자 인터페이스, 모니터링 정보, 통신방법 등에 대한 정보를 기술한다. 작성된 CMIML문서와 MAP파일을 이용해서 자동으로 소프트웨어를 생성함으로써 원격으로 제어되는 생산 현장의 여러 자동화 장비들에 적용될 수 있는 제어 및 모니터링 소프트웨어를 기존의 방법에 비해서 적은 비용으로 쉽고 빠르게 개발할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 국내외 연구동향에 대하여 살펴보고, 3장에서는 본 논문에서 제안하는 프레임워크에 대하여 세부적으로 설명한다. 끝으로 4장에서는 결론 및 향후 연구방향에 대해 살펴본다.

2. 관련 연구

국내에서는 소프트웨어의 개발보다는 자동화 장비의 구현이나 설비의 자동화, 또는 네트워크 프로토콜의 구현에 대한 연구가 더 활발히 진행되고 있다[1],[2]. 국외에서는 이미 오래 전부터 제어 및 모니터링 소프트웨어의 효율적인 개발에 대한 연구가 활발히 진행되어져 왔다. 그중 대표적인 것으로는 National Instrument사의 LabVIEW[3]와 Rockwell사의 RsVIEW[4], OPC(OLE for Process Control)[5], NASA의 AIML(Astronomical Instrument Markup Language)[6], 필드버스의 제어를 위해 제안된 다양한 논문들이 있다.

National Instrument사의 LabVIEW는 PC기반으로 장비들의 제어 및 모니터링을 위해 고안되었다. PC상에서 VI(Virtual Instument)를 구성할 수 있고 다양한 장비와 네트워크를 처리할 수 있도록 미리 많은 기능을 함수로 제공하고 있으며, 직접 코드를 작성하는 대신 다이어그램을 그리는 형식으로 프로그램을 하는 것이 특징이다. 주로 장비 또는 설비의 제어보다는 모니터링에 역점을 두고 있다.

Rockwell사의 RsVIEW는 LabVIEW와 마찬가지로 PC기반으로 장비들의 제어 및 모니터링을 위해 고안되었다. ActiveX나 OLE컨테이너 기능을 사용해서 RsVIEW의 기능을 확장할 수 있으며, 다양한 마법사와 도구들을 지원한다.

OPC는 장비와 제어 및 모니터링 소프트웨어 사이의 방대한 양의 데이터 통신을 지원할 수 있는 표준 기술로써 마이크로소프트사의 COM/DCOM 기술을 이용한 표준 인터페이스를 제공한다. OPC를 이용하게 되면 하드웨어 제공자는 OPC 표준을 따르는 단 하나의 클라이언트 컴포넌트만 제공하면 되고, 소프트웨어 개발자는 디바이스의 변경에 따라 소프트웨어를 다시 개발하지 않아도 되며, 사용자는 자동화 시스템의 구축 시 더 많은 선택을 할 수 있는 장점이 있다.

NASA의 AIML(Astronomical Instrument Markup Language)은 NASA에서 천문 관측에 사용하는 설비들의 제어를 위해서 개발한 명세이다. 소프트웨어와 하드웨어 개발자 사이에 명확한 인터페이스를 제공하고 XML 문서로 기술되며 Java와 함께 사용되어 플랫폼에 독립적인 특성을 제공한다.

필드버스의 제어를 위해 제안된 논문으로는 CAN필드버스의 편리하고 빠른 개발을 위해 고안된 Java CAN API 프레임워크를 소개하는 논문[7]과 CAN필드버스 시

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

스템을 XML 어플리케이션으로 표현하는 CoML을 제안하는 논문[8], CoML을 이용해서 CAN필드버스를 제어하는 CANINSIGHT시스템을 소개하는 논문[9], 그리고 필드버스 시스템의 동시 제어와 빠른 개발을 지원하는 JFCF(Java Fieldbus Control Framework)를 소개하는 논문[10]이 있다.

3. 제어 및 모니터링 소프트웨어 자동생성 프레임워크

본 논문에서 제안하는 프레임워크는 그림 1과 같이 세가지 표준문서 형식(ICD, MAP, CMIML), Instrument Description Wizard, Generator로 구성된다.

Instrument Description Wizard는 표준문서형식(ICD, MAP, CMIML)을 내장하고, 입력받은 ICD와 MAP파일을 이용해서 그림 2와 같이 GUI 기반으로 VI(Virtual Instrument)를 구성한다. 새롭게 구성된 VI를 이용하여 장비의 제어 정보, 사용자 인터페이스 정보, 모니터링 정보, 통신 방법 등에 대한 정보를 기술하는 CMIML을 생성한다.

Generator는 Instrument Description Wizard에서 생성한 CMIML문서와 실제 장비 드라이버에 대한 API를 입력으로 장비에 대한 제어 및 모니터링 소프트웨어를 자동 생성하게 된다.

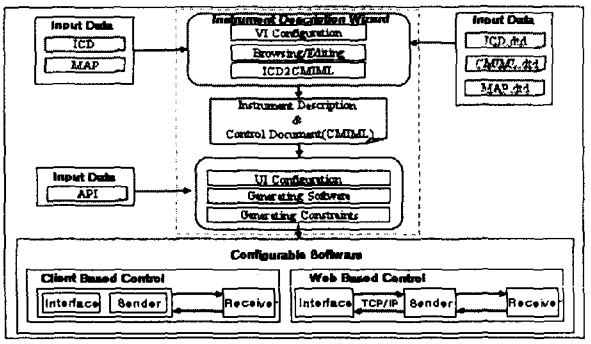

그림 1 CMIML Framework Architecture

3.1 ICD문서

장비 개발자는 장비의 동작정보와 상태 정보를 정의한 후 ICD문서를 작성한다. ICD는 유효한 XML문서이어야 하고 장비를 제어하기 위한 인터페이스정보(동작정보, 상태정보)를 기술해야한다. ICD의 최상위 요소는 Instrument이고 장비의 이름을 나타내는 Name과 장비에 대한 간단한 설명을 기술할 Desc속성을 가진다.

```
<!ELEMENT Instrument (FieldList?, MethodList)>
<!ATTLIST Instrument Name CDATA #REQUIRED
        Desc CDATA #REQUIRED>
```

장비의 상태정보를 표현하기 위한 요소로는 FieldList를 사용한다. 상태정보는 다수가 될 수 있기 때문에 FieldList는 하나 이상의 Field를 가질 수 있다. Field는 상태의 이름을 나타내는 Name속성과 간단한 설명을 기술하는 Desc속성을 가지고 하위 요소로는 상태의 데이터 타입을 기술할 Type요소와 상태값을 기술할 Value요소를 가진다.

```
<!ELEMENT FieldList (Field)+>
<!ELEMENT Field (Type, Value)>
<!ATTLIST Field Name CDATA #REQUIRED
        Desc CDATA #IMPLIED>
```

장비의 동작정보를 표현하기 위한 요소로는 MethodList를 사용한다. 동작정보는 다수가 될 수 있기 때문에 MethodList는 하나 이상의 Method를 가질 수 있다. Method는 동작의 이름을 나타내는 Name속성과 실행 후 반환하는 데이터의 타입을 나타내는 RetType속성, 간단한 설명을 기술하는 Desc속성을 가지고 하위 요소로는 동작 실행 시 필요한 파라미터를 기술할 ParamList요소를 가진다. ParamList요소는 다수의 Parameter 요소를 가질 수 있다. Parameter는 간단한 설명을 기술할 Desc속성을 가지고 하위 요소로는 파라미터의 이름과 데이터 타입을 기술할 Name과 Type요소를 가진다.

```
<!ELEMENT MethodList (Method)+>
<!ELEMENT Method (ParamList)?>
<!ATTLIST Method Name CDATA #REQUIRED
        RetType CDATA #REQUIRED
        Desc CDATA #IMPLIED>
<!ELEMENT ParamList (Param)+>
<!ELEMENT Parameter (Name, Type)+>
<!ATTLIST Parameter Desc CDATA #IMPLIED>
```

3.2 MAP문서

장비 드라이버 개발자는 장비의 API를 개발한 후 장비의 상태정보 및 동작정보와 API를 연결시키기 위해서 유효한 XML 문서인 MAP파일을 작성한다.

상태정보와 연결시키기 위한 요소로는 FieldMap요소를 사용한다. 다수의 상태정보가 존재할 수 있으므로 FieldMap은 하나 이상의 FieldReferenceMap을 가질 수 있다. FieldReferenceMap은 ICD에 기술되어 있는 상태정보를 나타내는 FieldReference속성과 실제로 연결될 API의 변수명을 나타내는 MapTo속성을 가진다.

```
<!ELEMENT FieldMap (FieldReferenceMap)+>
<!ELEMENT FieldReferenceMap ANY>
<!ATTLIST FieldReferenceMap
            FieldReference CDATA #REQUIRED
            MapTo CDATA #REQUIRED>
```

동작정보와 연결시키기 위한 요소로는 CommandMap요소를 사용한다. 다수의 동작정보가 존재할 수 있으므로 CommandMap은 하나 이상의 CommandReferenceMap을 가질 수 있다. CommandReferenceMap은 ICD에 기술되어 있는 동작정보를 나타내는 CommandReference와 실제로 연결될 API의 함수명을 나타내는 MapTo속성을 가지고, 함수가 가지는 파라미터 정보를 나타내기 위해서 ParamList요소를 하위요소로 가질 수 있다. ParamList는 다수의 파라미터가 존재할 수 있기 때문에 하나 이상의 Parameter를 가질 수 있다. Parameter는 파라미터의 데이터 타입과 값을 나타내기 위해서 Type과 Value요소를 하위요소로 가진다.

```
<!ELEMENT CommandMap (CommandReferenceMap)+>
<!ELEMENT CommandReferenceMap (ParamList)?>
<!ATTLIST CommandReferenceMap
            CommandReference CDATA #REQUIRED
            MapTo CDATA #REQUIRED>
<!ELEMENT ParamList (Parameter)+>
<!ELEMENT Parameter ANY>
<!ATTLIST Parameter Type CDATA #REQUIRED
            Value CDATA #REQUIRED>
```

3.3 CMIML 문서

Instrument Description Wizard를 통해서 생성되는 CMIML은 ICD의 확장이라고 볼 수 있다. ICD가 단순히 장비의 인터페이스정보만을 기술하고 있는 반면 CMIML은 ICD에 기술되어 있는 인터페이스정보에 제어 및 모니터링 소프트웨어의 사용자 인터페이스 정보와 장비의 모니터링 정보, 장비와 PC사이의 통신 방법, 장비의 스케줄 정보까지 기술하고 있다.

CMIML의 최상위 요소는 Instrument이고 이름을 나타내는 Name속성과 간단한 설명을 기술할 Desc속성을 가지고 있다. 설비는 하나의 장비로 구성될 수도 있지만 여러 장비가 모여 구성될 수도 있다. 따라서 Instrument 요소는 다시 Instrument를 하위요소로 가질 수 있다. 그리고 하나의 장비는 통신 방법을 정의하는 Port요소와 장비의 스케줄 정보를 나타내는 ScheduleList요소를 하위 요소로 가진다.

```
<!ELEMENT Instrument
        (Instrument | (Port, ScheduleList?))*>
<!ATTLIST Instrument Name CDATA #REQUIRED
        desc CDATA #IMPLIED>
```

장비와 PC사이의 통신정보를 나타내기 위해서는 Port요소를 사용한다. Port요소는 어떤 장비의 포트인지를 나타내는 Name속성과 데이터를 주는 포트인지 받는 포트인지를 구분하는 Function속성(Command, Response), 포트번호를 나타내는 Number속성, 포트를 통해서 전송되어지는 데이터의 타입을 정의하는 Type속성

(Ascii, Binary), 그리고 다른 네트워크의 장비일 경우 유일한 이름을 식별하는 Hostname속성을 가진다. 하위 요소로는 장비의 제어정보를 나타내는 Command와 Field요소를 가질 수 있고, 조건문이나 반복문과 같은 제어 정보를 나타내는 Control 요소를 가질 수 있다.

```
<!ELEMENT Port ((Command|Control)+|(Field)*)>
<!ATTLIST Port Name CDATA #REQUIRED
        Function CDATA #REQUIRED
        Number CDATA #REQUIRED
        Type CDATA #REQUIRED
        Hostname CDATA #IMPLIED>
```

CMIML이 ICD와 다른 점 중 하나는 사용자 인터페이스 정보를 가지고 있다는 것이다. 장비의 제어정보가 사용자에게 표시될 지 아니면 스케줄의 일부인지를 결정하는 정보를 가지고 있어야 한다. 기존의 DTD에 다음의 속성을 추가한다.

```
UI CDATA #REQUIRED
UI_ID CDATA #IMPLIED
```

CMIML의 대표적인 Control요소로는 If, While, For문 등이 있다. If와 While문은 조건을 표시하는 Condition속성만을 가지며 For문의 경우는 조건식을 State1, State2, State3 속성으로 가진다. Control요소의 Id속성은 유일한 Control요소를 식별하기 위해 사용되어 진다.

```
<!ELEMENT Control
        ((If,(ElseIf)*,Else)|(If,(ElseIf)*)|While|For)>
<!ATTLIST Control Id CDATA #REQUIRED>
<!ELEMENT If (Command | Control)*>
<!ATTLIST If condition CDATA #REQUIRED>
<!ELEMENT ElseIf (Command | Control)*>
<!ATTLIST ElseIf condition CDATA #REQUIRED>
<!ELEMENT Else (Command | Control)*>
<!ELEMENT While (Command | Control)*>
<!ATTLIST While condition CDATA #REQUIRED>
<!ELEMENT For (Command | Control)*>
<!ATTLIST For State1 CDATA #REQUIRED
              State2 CDATA #REQUIRED
              State3 CDATA #REQUIRED>
```

CMIML에서는 ScheduleList요소를 사용해서 장비에 대한 일련의 동작을 기술하는 스케줄 정보를 저장한다. 한 장비가 다수의 스케줄을 수행할 수 있기 때문에 ScheduleList요소는 0개 이상의 Schedule요소를 가질 수 있다. Schedule요소는 스케줄의 이름을 나타내는 Name속성과 사용자 인터페이스 여부를 나타내는 UI속성(True, False)과 스케줄을 시작하는 인터페이스를 구별할 UI_IDS와 스케줄을 종료하는 인터페이스를 구별하는 UI_IDE속성을 가진다.

```
<!ELEMENT ScheduleList (Schedule)*>
<!ELEMENT Schedule (Command|Control|Schedule)+>
<!ATTLIST Schedule Name #REQUIRED
        UI CDATA #REQUIRED
        UI_IDS CDATA #IMPLIED
        UI_IDE CDATA #IMPLIED>
```

3.4 Instrument Description Wizard

Instrument Description Wizard는 표준문서형식(ICD, MAP, CMIML)을 내장하고, 장비에 대한 인터페이스정보를 기술한 ICD파일과 ICD의 인터페이스정보와 실제 API나 Address Map과의 연결정보를 기술한 MAP파일을 이용해서 GUI 기반으로 VI(Virtual Instrument) 편집 환경을 제공하는 도구이다. 입력받은 ICD에서 장비의 동작정보만 추출해서 그림 2와 같이 좌측의 Standard ICD 메뉴에 추출된 동작 리스트가 표시된다.

새로운 VI를 생성하기 위해서는 좌측 리스트의 한 아이템을 드래그&드롭으로 우측의 Design탭에 올려놓고, 파라미터, 속성, 조건 등을 설정하고, 각 항목들을 연결한다. 새롭게 생성된 VI는 다른 VI의 구성에 사용되어질 수 있다.

3.5 Software Code Generator

Instrument Description Wizard를 이용해서 생성한 CMIML을 입력으로 받고, 사용자 인터페이스를 편집한 후 제어 및 모니터링 소프트웨어 및 실시간 제약사항(Realtime Constraints)을 생성하는 도구이다.

생성된 인터페이스들은 고유의 ID를 가지고 CMIML의 각 요소의 UI_ID속성과

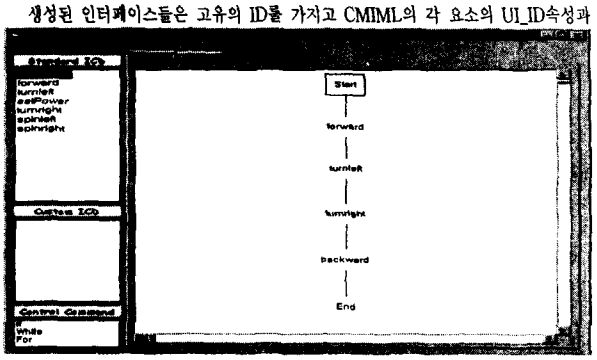

그림 2 Virtual Instrument Editing

연결되고, CMIML에 기술된 Field와 Command 요소들은 MAP파일을 참조해서 실제 API로 생성된다.

실시간 제약사항은 소프트웨어 실행시에 참조해서 실행 가능한 요소들을 추출한 것으로써 소스코드의 변경없이 실행을 변경할 수 있도록 한다.

4. 결론 및 향후 연구과제

본 논문에서는 생산현장의 다양한 장비와 설비들의 제어 및 모니터링 소프트웨어에 대한 쉬운 개발과 유지보수성을 향상시킬 수 있는 프레임워크를 제안하였다.

장비에 대한 인터페이스정보를 기술하는 ICD문서와 ICD의 인터페이스와 실제 API의 연결 정보를 기술하는 MAP문서를 작성하기 위한 DTD를 정의하였다. 그리고 장비의 제어정보 및 사용자 인터페이스 정보, 모니터링 정보, 장비와 PC사이의 통신방법, 장비의 스케줄링 정보를 저장하는 CMIML문서의 DTD를 정의하였다.

제안한 프레임워크는 GUI 기반으로 VI를 구성하고 CMIML의 생성을 지원하는 Instrument Description Wizard와 CMIML을 이용해서 장비에 대한 제어 및 모니터링 소프트웨어를 생성하는 Software Code Generator로 구성되었다.

향후 연구과제로는 소프트웨어 코드의 변경을 최소화하고 무정지 시스템을 가능하게 하는 실시간 참조 제약사항에 대한 정의와 자동 생성된 소프트웨어의 안정성을 검증할 수 있는 가상 시뮬레이션에 대한 연구가 필요하다.

[참고문헌]

[1] 김지용, 홍승호, 김기암, "PROFIBUS FMS를 이용한 파이프 밴딩 공정 자동화", 제어계측,자동화,로보틱스연구회합동학술발표회, 1997

[2] 이철민, 이재환, 장태정, 남부희, "PROFIBUS Protocol 및 인터페이스 H/W의 구현", 제어계측,자동화,로보틱스연구회합동학술발표회, 1997

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

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

[5] OPC "www.opcfoundation.org/"

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

[7] Buhler Dieter, Gerd Nusser, "The Java CAN API A Java Gateway to Fieldbus Communication", IEEE, 2000

[8] Buhler Dieter, "The CANOpenn Markup Language Representing Fieldbus Data with XML", IEEE, 2000

[9] Buhler Dieter, Kuchlin Wolfgang, "Remote Fieldbus System Management with Java and XML", IEEE, 2000

[10] Buhler Dieter, Gerd Nusser, Kuchlin Wolfgang, Gruhler Gerhard, "The Java Fieldbus Control Framework Object oriented control of fieldbus devices", IEEE, 2001

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