필드기기 통합구성을 위한 XML 적용에 관한 연구

A Study about XML Application for Integrating Field Device

문 용 선*, 이 명 복, 정 철 호
(Yong Seon Moon, Myung Bok Lee, and Cheol Ho Jeong)

Abstract : Device Description that is described as C-based *DDL*(Device Description Language) should be provided for the control system to use the progressive function of field device in the distributed control system. DD expresses field device function itself and parameter. DD is used to set a necessary function for a control strategy in the central control system. DD technology which was adopted as international standard IEC 61804-Part 2 is being used in PROFIBUS, Foundation Fieldbus and HART. However, the DD is dependent to the fieldbus and it doesn't have a suitable form in the industrial Ethernet. This paper presents the possibility of a change from EDD(Electronic Device Description) and GSD(General Slave Data) used for integrating field devices to XDD(XML for Device Description) and verifies it's effectiveness. Also, this paper presents research assignment on the XML application in the distributional system from now on by examining the possibility of Ethernet's development in the industrial area.

Keywords : fieldbus, integrating field device, EDD, XML

I. 서론

산업현장의 필드기역에 위치하는 필드기기를 중앙제어시스템에 통합구성하기 위해서는 필드기기의 논리적 구조 및 기능을 표현하는 DD(Device Description)를 중앙제어시스템에 제공해야 한다. DD는 C언어 기반의 DDL(Device Description Language)로 기술되어 있으며, 현재 국제 표준인 IEC 61084-Part 2로 채택되어 있고, 필드버스 기술인 PROFIBUS, Foundation Fieldbus, HART에서 사용하고 있다.

그러나 DD 기술은 필드기기의 자체 기능 및 파라미터의 정보를 중앙제어시스템에 제공해 줄 수 있어 제어시스템의 기능적 분산을 가능하게 하였지만, 필드버스 기술에 종속적이어서, 서로 다른 필드버스에서 구성된 필드기기들을 단일 제어시스템으로 통합할 수 없다. 또한 산업용 제어 네트워크로 빠른 성장을 보이고 있는 산업용 이더넷[1-4]과 함께 웹 기반 어플리케이션에서 사용하기 위한 표준화된 데이터 형식을 갖추고 있지 않다. 따라서 필드기기의 자체 기능 및 파라미터를 표현하는 DD 기술을 필드버스에 종속적이지 않으면서, 산업용 이더넷환경에서도 적합한 데이터 형식을 갖춰야 한다.

이러한 요구사항으로 인해 현대의 제어시스템에서 DD를 이더넷환경의 웹 기반 어플리케이션에서 표준화되고, 필드버스 기술에도 종속되지 않는 XML(Extensible Markup Language) 을 이용하여 공통적인 정보모델과 구조를 기반으로 필드기기의 기능 및 파라미터를 표현하기 위한 노력이 진행되고 있다[5-13].

따라서 본 논문에서는 필드버스 기술인 PROFIBUS에서 사용되는 EDD(Electronic Device Description)와 GSD(General Slave Data)을 XML로 표현하여 공통적인 정보모델과 구조를 제시한다. 또한 앞으로 산업현장에서 이더넷의 발전 가능성을 살펴봄으로써, 산업현장의 제어시스템에서의 XML 적용한 향후 연구 과제를 제시한다.

* 책임저자(Corresponding Author)
논문접수 : 2004. 1. 19., 채택확정 : 2005. 3. 18.
문용선, 이명복, 정철호 : 순천대학교 전자공학과
(yongseon@sunchon.ac.kr/blues715@naver.com/jch@poscon.co.kr)

II. 분산 환경에서의 필드기기와 필드기기의 표현

1. 제어시스템의 기능적 분산

산업현장의 분산 제어시스템 구축을 가능하게 하는 것은 실시간 네트워크 기술인 필드버스와 지능형 필드기기의 발달로 가능하게 되었다. 필드버스의 실시간 통신 네트워크를 이용하여 자체적인 기능을 탑재하고 있는 지능형 필드 기기는 내부 파라미터 및 상태, 프로세스 값들의 온라인 전송과 진단, 실시간 모니터링 기능을 제공한다[14]. 그림 1은 필드버스와 지능화된 필드기기의 사용으로 인해 저수준 제어, 지능 불록 실행, 알람 처리 등 여러 기능들이 필드 지역으로 이동되어 있는 나타내고 있다[15].

따라서 제어시스템의 기능적 분산으로 인해 필드기기의 자체 파라미터 및 기능들을 표현하는 DD 기술이 발전하게 되었다.

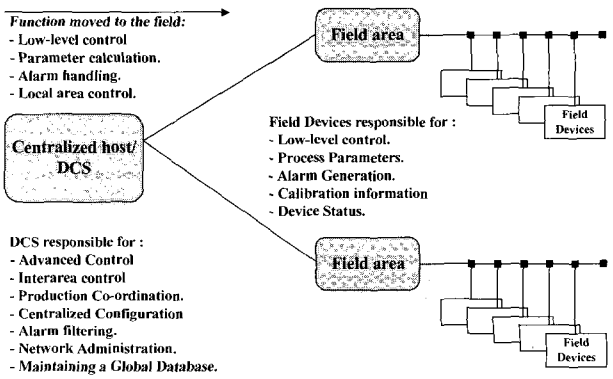


그림 1. 필드 지역으로 분산된 제어기능.
Fig. 1. Distributed control function to field area.

표 1. 필드기기의 기본 표현 요소.
Table 1. Basic description element of field device.

기본 표현 요소	설 명
데이터(변수 및 파라미터) 표현	프로세스 변수 - 변수 타입, 접근 권한, 기본값 등.
Human Interface support	데이터 설명을 위한 텍스트, 소프트웨어 툴의 메뉴 구성 등.
필드 기기 구성 파라미터	필드 기기 식별자 등.
통신 관련 파라미터	네트워크 주소, 전송률 = 500kBaud
기 타	기능 블록 구성 및 스케쥴링.

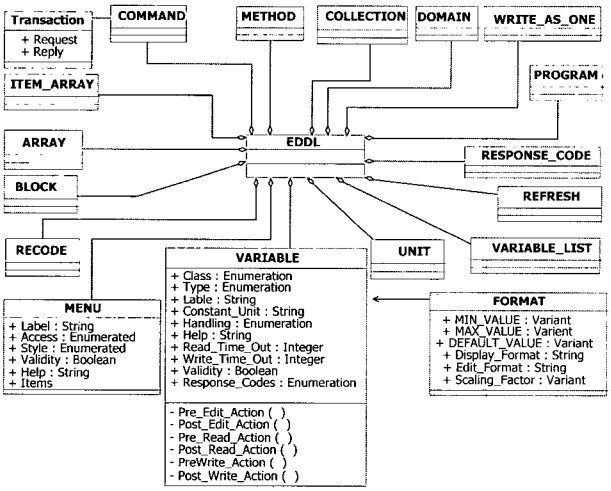


그림 2. PROFIBUS EDDL의 UML 표현.
Fig. 2. PROFOFIBUS EDDL's UML expression.

2. 필드기기의 표현

DD의 가장 중요한 목적은 중앙 제어시스템에서 다양한 필드기기를 통합 구성할 수 있도록 통신 능력에 관한 정보와 기기의 기능, 사용자 인터페이스를 위한 파라미터 및 변수 등을 정보를 제공해야 한다. 이러한 정보는 공통된 정보 구조 및 모델을 기반으로 표현되며, 필드기기간, 네트워크 계층간 정보의 호환성을 제공할 수 있는 중요한 요소이다. 필드기기의 기본 표현 요소를 표 1에 나타내었다[9,10].

DD 기술은 IEC 61084-Part 2로 제정되어 있으며, PROFIBUS, Foundation Fieldbus, HART에서 사용되고 있으며, 그림 2는 PROFIBUS에서 사용하는 EDD 기본 요소를 UML(Unified Modeling Language)의 클래스 다이아그램(class diagram)으로 표현하였다. 또한 세부적인 속성이나 동작은 정의하지 않았고, 구성 요소들의 집합 관계만 표시하였다.

3. 필드기기 표현의 문제점

현재의 DD 기술이 국제 표준으로 채택되어 공통적인 정보 모델과 구조를 제공하고 있지만, 필드버스 기술에 따라 표현 방법과 구조가 서로 상이함에 따라 전체 제어시스템을 통합 구성하기 위해 추가적인 시간과 비용이 요구된다. 이것은 필드기기의 선택폭이 줄어 들 수밖에 없으며, 사용자가 원하는 진정한 개방형 제어시스템을 구현할 수 없다. 또한, 현재의 산업용 이더넷은 네트워크 계층의 제어계층과 필드 계층 사이에 존재하는 원격 입출력 모듈까지 적용범위를 넓혀가고 있으며, 이와 더불어 산업용 이더넷 환경의 웹 기반 어플리케이션에서 산업현장의 데이터를 관리하고 처리하고자 하지만, 현재의 DD 기술은 이더넷 환경의 웹 기반 어플리케이션에서 표준화된 데이터 형식을 갖추고 있지 않다.

따라서 이러한 문제를 해결하기 위해서는 필드버스 기술에 독립적이어야 하며, 산업용 이더넷 환경에서의 웹 기반 어플리케이션에서 사용하기 적합한 데이터 형식을 갖추어야 한다. 또한 기존의 공통적인 정보 모델과 구조를 제공하기 위해 필드기기 표현의 유효성 검사가 가능해야 한다.

III. 필드기기 표현을 위한 XML과 XML를 이용한 필드기기의 표현 구조 설계

1. 필드기기 표현을 위한 XML

XML은 웹 상에서 구조화 된 문서를 전송 가능하도록 설계되었고, 또한 W3C(World Wide Web Consortium)에서 14개의 회사나 기관이 참여하여 정의한 기술로, 현재 DD 기술의 문제점을 해결할 수 있는 여러 가지 기술을 포함하고 있다. XML의 가장 두드러진 특징은 미리 정의된 요소가 없고, 문서의 스타일정보와 데이터 구조를 분리하여 전송함으로써, 데이터베이스의 정렬과 전송을 쉽고 빠르게 할 수 있다는 것이다.

그 중 필드 기기의 표현을 위한 XML 기술은 다음과 같이 3가지로 나눌 수 있다. 첫째, 사용자에게 데이터를 표현할 수 있는 XSL(Extensible Stylesheet Language). 둘째, 상위 어플리케이션과 상호 인터페이스와 동작을 구현하기 위한 DOM(Document Object Modeling)과 SAX(Simple API for XML). 셋째, 필드기기 정보가 표현된 XML 파일의 공통적인 정보 모델과 구조를 제공함으로써, 문서의 유효성을 검증할 수 있는 DTD(Document Type Definition)와 스키마(Schema) 이다. 그림 3은 상위 어플리케이션과 XML 데이터와의 상호작용을 나타낸 그림으로, 상위 어플리케이션에서는 DOM과 SAX를 통해, XML 구조에 접근하고, XML 데이터 표현은 XSLT, 구조화된 문법은 DTD, 스카마로 정의하고 있다.

그림 4는 기존의 필드기기 표현을 XML로 표현하면, XSL 스타일시트 규칙에 의해 사용자가 필요로 하는 데이터만을 보여줄 수 있도록 표현할 수 있으며, 기존의 필드기기 표현 파일을 생성할 수 있다. 예를 들어 foundation fieldbus에서는 CFF, PROFIBUS는 GSD을 생성할 수 있는 것으로, 기존의 기술을 그대로 사용할 수 있는 장점과 새로운 기술에 대한 신뢰성을 제공해 준다.

2. 필드기기의 표현 구조 설계

필드기기의 표현 구조를 설계하기 위해서 본 논문에서는 공통적인 정보모델과 구조로, XML를 이용한 XDD(XML for Device Description)로 표현한다. XDD의 구조의 그림 5에서 제시하였다.

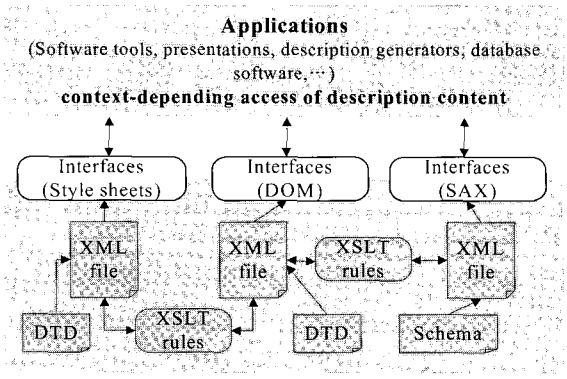


그림 3. 상위 어플리케이션의 XML 접근과 DTD, XSL의 상호 작용.

Fig. 3. XML access of application and interaction of DTD, XSL.

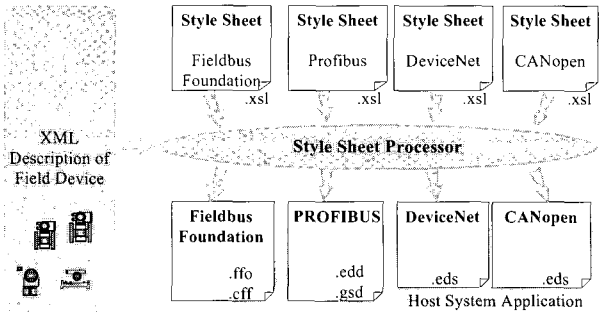


그림 4. 스타일시트를 이용한 DD파일 생성.

Fig. 4. DD file creation using style sheet.

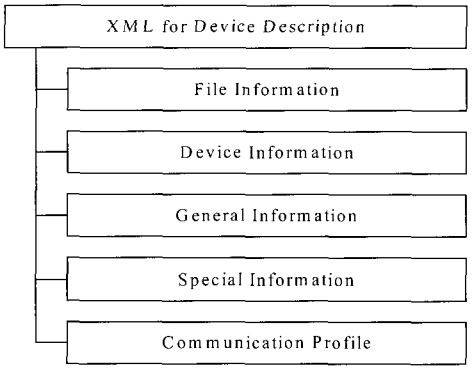


그림 5. XDD 구조.

Fig. 5. XDD structure.

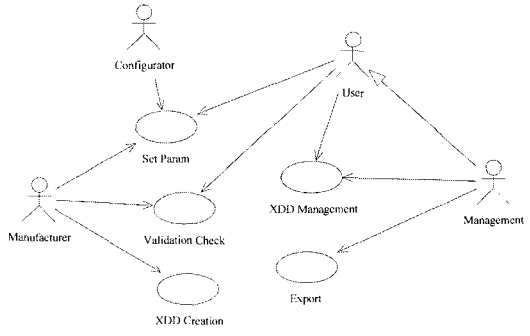


그림 6. VRLab XDD 유저케이스.

Fig. 6. VRLab XDD - Use case.

그림 5에서 최상위 요소를 나타내는 XML for Device Description과 하위 요소인 파일정보를 나타내는 File Information, 기기 자원을 나타내는 Device Information, 필드기기의 공통적인 정보를 나타내는 General Information, 필드기기의 생산자 특정 기능을 나타내는 Special Information, 각각의 필드버스 통신 특성에 맞는 파라미터를 나타내는 Communication Profile로 구성된다.

필드기기의 기능 및 파라미터를 XDD로 표현하기 위해 본 논문에서는 자바 어플리케이션인 VRLab XDD를 구현하였으며, 그림 6에서 기능적 측면을 표현하였다.

Ⅳ. XML를 이용한 필드기기 표현

1. 필드기기 표현을 위한 DTD

XDD 구조를 정형화된 형태로 기술하기 위해 DTD로 표현한 그림 7에서는 최상위 요소인 XDD와 XDD 하위요소에 표 1에서 나타낸 필드기기의 정보를 표현할 수 있도록 하였다.

그림 7에서 CommProfile 요소의 파라미터는 Foundation Fieldbus와 PROFIBUS, HART 경우에 서로 다른 통신 설정에 필요한 정보를 포함하고 있으며, 현재 Foundation Fieldbus에서는 CFF(Common File Format), PROFIBUS에서는 GSD(General Slave Data) 파일의 내용을 포함하게 된다.

그림 8은 필드기기 표현에서 가장 많은 부분을 차지하는 GeneralSpec 요소 하위의 Variable 구조를 DTD로 작성하였다. Variable 요소에서 필수적인 속성은 DTD 문법에 따라 #REQUIRED 로 정의하였고, 이외의 속성은 #IMPLIED 로 정의하였다. 또한, 속성의 추가적인 정보와 조건문은 Variable 하위 요소인 VARIABLE_SECTION 요소로 구조화하였으며, 그림 9는 사용자 인터페이스 부분을 담당하는 Menu 요소를 DTD로 작성한 그림이다.

필드버스 통신에 관한 정보를 포함하는 CommProfile DTD는 그림 10에서 구조화 하였으며, CommProfile 하위에 포함되는 PROFIBUS 통신 특성을 포함하여 작성한 DTD 구조의 일부이다. GeneralKey 속성에서 GSDRevision, Vendor Name, ModelName, Revision 이외에 기기 정보와 지원되는 통신속도, 응답시간 등을 정의한다. Hardware Key 요소에는 PROFIBUS 물리계층에서의 인터페이스 특성과 전송지연시간 등을 정의하고, PROFIBUS 통신에서 마스터 기능을 지원하기 위한 Master Key 요소, 슬레이브 기능을 지원하기 위한 Slave Key 요소를 정의하였다.

```
<!ELEMENT XDD (FileInfo, DeviceInfo, GeneralSpec, SpecialSpec,
CommProfile)>

<!ELEMENT FileInfo (XDDRevision)>
<!ELEMENT DeviceInfo (Manufacturer, DeiveType, DeviceRevision)>
<!ELEMENT GeneralSpec (Variable*, Menu*, Unit*, VariableList*,
Refresh*, Program*, Domain*, Collection*, Method*, Command*,
ItemArray*, Array*, Block*, Recode*, WriteAsOne*,
ResponseCode*)>
<!ELEMENT CommProfile (FoundationFieldbus | PROFIBUS |
HART)>
```

그림 7. XDD - DTD.

Fig. 7. XDD - DTD.

```
<!-- ===== Start Valiable Element Declaration ===== -->
<!ELEMENT Variable (VARIABLE_SECTION)>
<!ATTLIST Variable
        Name CDATA #REQUIRED
        Class CDATA #REQUIRED
        Type CDATA #REQUIRED
        Label CDATA #REQUIRED
        ConstantUnit CDATA #IMPLIED
        Handling CDATA #IMPLIED
        Help CDATA #IMPLIED
        ReadTimeOut CDATA #IMPLIED
        WriteTimeOut CDATA #IMPLIED
        Validation CDATA #IMPLIED
        ResponseCodes CDATA #IMPLIED
        PreEditAction CDATA #IMPLIED
        PostEditAction CDATA #IMPLIED
        PreReadAction CDATA #IMPLIED
        PostReadAction CDATA #IMPLIED
        PreWhiteAction CDATA #IMPLIED
        PostWriteAction CDATA #IMPLIED>
<!ELEMENT VARIABLE_SECTION (Class?, Type?, Label?,
ConstantUnit?, Handling?, Help?, ReadTimeOut?, WriteTimeOut?,
Validation?, ResponseCodes?, PreEditAction?, PostEditAction?,
PostReadAction?, PreWhiteAction?, PostWriteAction?)>
<!ELEMENT Class (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT Label (#PCDATA)>
<!ELEMENT ConstantUnit (#PCDATA)>
<!ELEMENT Handling (#PCDATA)>
<!ELEMENT Help (#PCDATA)>
<!ELEMENT ReadTimeOut (#PCDATA)>
<!ELEMENT WriteTimeOut (#PCDATA)>
<!ELEMENT Varidation (#PCDATA)>
<!ELEMENT ResponseCodes (#PCDATA)>
<!ELEMENT PreEditAction (#PCDATA)>
<!ELEMENT PostEditAction (#PCDATA)>
<!ELEMENT PostReadAction (#PCDATA)>
<!ELEMENT PreWhiteAction (#PCDATA)>
<!ELEMENT PostWriteAction (#PCDATA)>
```

그림 8. Variable DTD.
Fig. 8. Variable DTD.

```
<!-- ====== Start Menu Element Declaration ====== -->
<!ELEMENT Menu (MENU_SECTION)>
<!ATTLIST Menu
        Name CDATA #REQUIRED
        Label CDATA #IMPLIED
        Item CDATA #IMPLIED
        Access CDATA #IMPLIED
        Style CDATA #IMPLIED
        Help CDATA #IMPLIED
        Validity CDATA #IMPLIED
>
<!ELEMENT MENU_SECTION (Item)>
<!ELEMENT Item (#PCDATA)>
```

그림 9. Menu DTD.
Fig. 9. Menu DTD.

2. 필드기기 표현을 위한 스타일시트

작성된 DTD 구조에 따라 표현된 XDD의 정보가 사용자에게 시각적인 정보로 제공하기 위해서 스타일 시트 구조를 그림 11과 같이 정의하였으며, HTML의 데이터 형식을 갖추고 있다.

```
<!ELEMENT CommProfile (PROFIBUS | FoundationFieldbus |
DeviceNet)>
<!ATTLIST CommProfile
        type (PROFIBUS | FoundationFieldbus | DeviceNet)
#REQUIRED>
<!ELEMENT PROFIBUS (GeneralKey, HardwareKey?, MasterKey?,
SlaveKey?)>
<!ELEMENT GeneralKey EMPTY>
<!ATTLIST GeneralKey
        GSDRevision CDATA #REQUIRED
        VendorName CDATA #REQUIRED
        ModelName CDATA #REQUIRED
        Revision CDATA #REQUIRED
        RevisionNumber CDATA #IMPLIED
        IdentNumber CDATA #REQUIRED
        ProtocolIdent CDATA #REQUIRED
        StationType (0 | 1) "0"
        FMSsupp (0 | 1) "0"
        HardwareRelease CDATA #REQUIRED
        SoftwareRelease CDATA #REQUIRED
        CommSupp_9.6 (0 | 1) "0"
        CommSupp_19.2 (0 | 1) "0"
        CommSupp_31.25 (0 | 1) "0"
        CommSupp_45.45 (0 | 1) "0"
        CommSupp_93.75 (0 | 1) "0"
        CommSupp_187.5 (0 | 1) "0"
        CommSupp_500 (0 | 1) "0"
        CommSupp_1.5M (0 | 1) "0"
        CommSupp_3M (0 | 1) "0"
        CommSupp_6M (0 | 1) "0"
        CommSupp_12M (0 | 1) "0"
        ResponseTime_9.6 CDATA #REQUIRED
        ResponseTime_19.2 CDATA #REQUIRED
        ResponseTime_31.25 CDATA #REQUIRED
        ResponseTime_45.45 CDATA #REQUIRED
        ResponseTime_93.75 CDATA #REQUIRED
        ResponseTime_187.5 CDATA #REQUIRED
        ResponseTime_500 CDATA #REQUIRED
        ResponseTime_1.5M CDATA #REQUIRED
        ResponseTime_3M CDATA #REQUIRED
        ResponseTime_6M CDATA #REQUIRED
        ResponseTime_12M CDATA #REQUIRED
        Redundancy (0 | 1) "0"
        RepeaterCtrlSig (0 | 1) "0"
        Voltage24VPins (0 | 1) "0"
        ImplementationType CDATA #IMPLIED
        BitmapDiag CDATA #IMPLIED
        BitmapDevice CDATA #IMPLIED
        BitmapSF CDATA #IMPLIED>
```

그림 10. CommProfile - PROFIBUS DTD.
Fig. 10. CommProfile - PROFIBUS DTD.

그림 12는 XDD에서 GSD 파일을 생성하기 위한 스타일 시트이며, 템플리트 구조에서 GSD 파일정보를 포함하고 있는 CommProfile 하위요소인 PROFIBUS 요소의 값들을 출력하게 된다. <xsl:output method="text"/>로 정의하여 텍스트 형식으로 출력하듯이 DD 파일의 생성 또한 같은 방법으로 XSLT, XPath를 이용하여 텍스트 형식으로 생성한다.

3. VRLab XDD 구현

그림 13는 구현된 VRLab XDD를 이용하여 그림 8에서 정의한 Variable 요소를 생성하는 화면과 생성된 Variable 요소의 일부이며, 웹 브라우저에서 보여준 화면이다. 또한 그림 14는 Menu 요소를 생성하기 위한 화면과 생성된 Menu 일부분을 표시하였다.

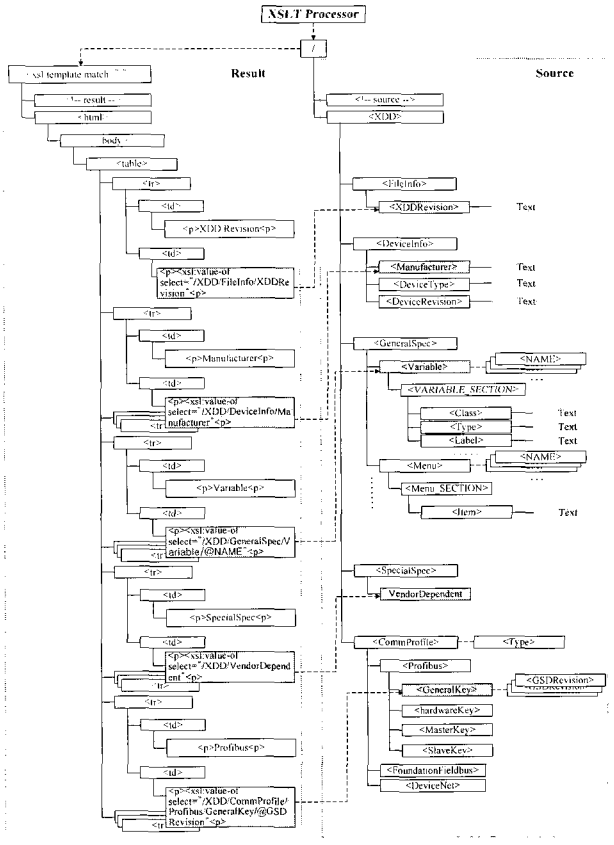


그림 11. XDD 스타일 시트 구조.

Fig. 11. Structure of XDD style sheet.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:template match="/">
;************************************************************
;**          GSD file for                                   **
;**          Manufacture : VRLab                            **
;**          Demo.GSD                                       **
;************************************************************
#PROFIBUS_DP
 <xsl:apply-templates/>
 </xsl:template>
 <xsl:template match="CommProfile/PROFIBUS">
GSD_Revision =<xsl:value-of select="GeneralKey/@GSDRevision"/>
Vendor_Name =<xsl:value-of select="GeneralKey/@VendorName"/>
Model_Name = <xsl:value-of select="GeneralKey/@ModelName"/>
Revision     =<xsl:value-of select="GeneralKey/@Revision"/>
Ident_Number = <xsl:value-of select="GeneralKey/@IdentNumber"/>
Protocol_Ident= <xsl:value-of select="GeneralKey/@ProtocolIdent"/>
Station_Type = <xsl:value-of select="GeneralKey/@StationType"/>
FMS_supp = <xsl:value-of select="GeneralKey/@FMSSupp"/>
Hardware_Release = <xsl:value-of select="GeneralKey/@HardwareRelease"/>
Software_Release =<xsl:value-of select="GeneralKey/@SoftwareRelease"/>
9.6_supp     = <xsl:value-of select="GeneralKey/@CommSupp_9.6"/>
19.2_supp = <xsl:value-of select="GeneralKey/@CommSupp_19.2"/>
31.25_supp = <xsl:value-of select="GeneralKey/@CommSupp_31.25"/>
MaxTsdr_9.6 = <xsl:value-of select="GeneralKey/@ResponseTime_9.6"/>
MaxTsdr_19.2 = <xsl:value-of select="GeneralKey/@ResponseTime_19.2"/>
 </xsl:template>
</xsl:stylesheet>
```

그림 12. GSD 파일 생성을 위한 스타일 시트.

Fig. 12. Style sheet for GSD file creation.

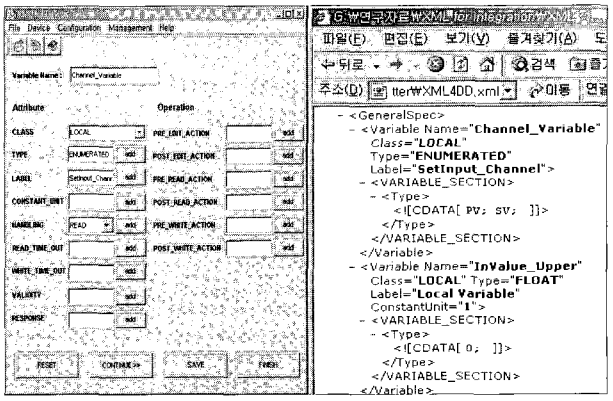


그림 13. DOM를 이용한 variable 생성.

Fig. 13. Variable creation using DOM.

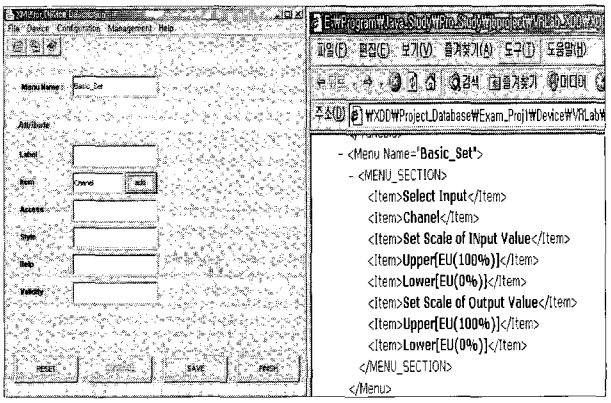


그림 14. DOM를 이용한 Menu 생성.

Fig. 14. Menu creation using DOM.

그림 15. XMLSPY를 이용한 XDD 유효성 검사.

Fig. 15. XDD validation check using XMLSPY.

4. DTD를 이용한 유효성 검사와 파라미터 설정

그림 15에서는 그림 7에서 작성된 DTD를 이용하여 생성된 XDD의 공통적인 정보모델과 구조의 유효성을 검사한화면이다. 유효성을 판단하기 위해 XMLSPY[16]를 이용하였

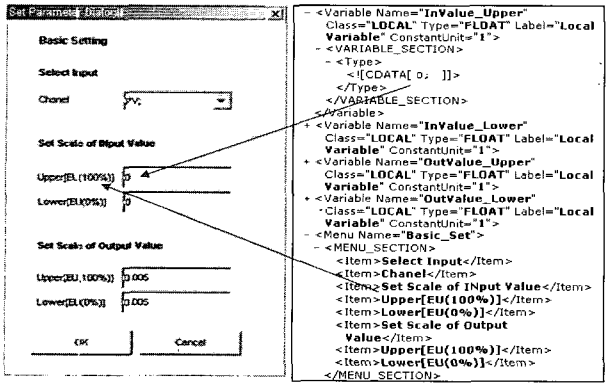


그림 16. 필드기기 파라미터 설정

Fig. 16. Field device parameter configuration

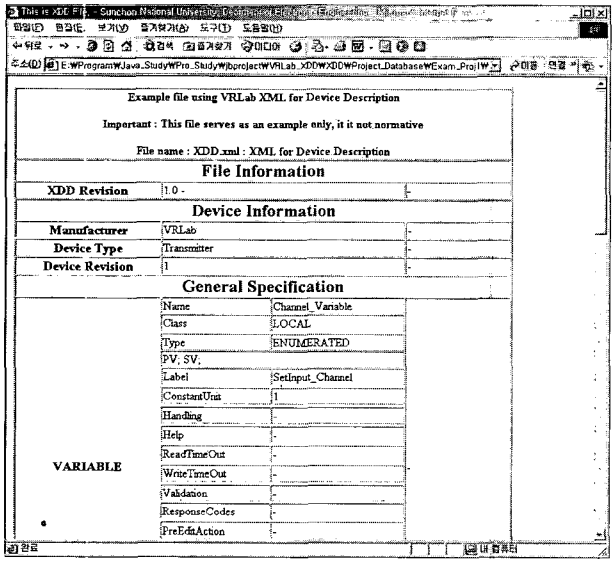


그림 17. XDD의 스타일 시트 적용.

Fig. 17. Style sheet application of XDD.

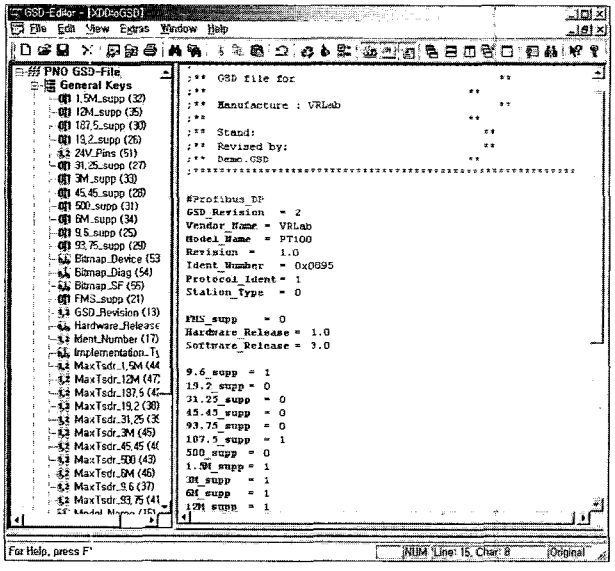


그림 18. XDD를 이용한 GSD 파일의 생성.

Fig. 18. GSD file creation using XDD.

으며, 중앙 하단부에 "This is a Valid"라는 문구가 유효성을 판단한 것이다. 이것은 필드기기 정보의 공통적인 정보 모델과 구조를 제공해 줄 수 있다는 측면에서 매우 중요하다. 그림 16은 사용자 측면에서 XDD의 데이터를 이용하여 파라미터를 설정하는 화면이다. 그림 13, 14에서 생성한 Variable과 Menu의 데이터를 DOM 으로 접근하여 파라미터를 설정할 수 있는 다이얼로그 화면에 표시하였다.

그림 17은 생성된 XDD를 그림 11의 스타일시트를 적용하여 브라우저를 통해 나타낸 화면이다.

본 논문에서는 테이블 내에 XDD의 정보를 표시하였지만, 스타일시트는 사용자가 원하는 모양으로 출력이 가능하다. PROFIBUS의 통신 및 네트워크 설정에 관한 정보를 정의하는 GSD 파일을 그림 4와 같은 방법으로 생성하였으며, 그림 12에서 정의한 스타일시트를 이용하여 출력해 PROFIBUS 협회에서 제공하는 GSD Editor를 이용하여 정보를 표시하였다. 이것은 기존의 GSD 파일의 구조와 동일하게 생성할 수 있다는 것을 나타낸다.

V. 결론

본 논문에서는 XML를 이용하여, 필드기기 표현에 필요한 공통적인 정보모델과 구조를 XDD로 생성하였다. 또한, 필드버스에 종속적이지 않고 구조화된 XDD의 생성과 유효성 검사를 위한 DTD를 정의하였으며, 정의된 DTD 구조에 맞게 Variable 요소와 Menu 요소를 생성하여 파라미터를 설정하고, 생성된 XDD를 스타일시트를 적용하여 사용자에게 표현할 수 있는 방법의 한가지로 웹 브라우저에서 테이블로 사용자에게 쉽게 표현하였다. 마지막으로 스타일시트를 적용하여 현재 사용되고 있는 GSD 파일을 생성하여 현재 DD의 정보를 모두 포함할 수 있는 구조로 되어 새로운 기술에 대한 신뢰성을 제공할 수 있다.

필드 디바이스의 표현을 XML로 표현하게 되면, 필드기기의 기능 및 파라미터뿐만 아니라 구성 소프트웨어에 따른 의존적인 구성데이터가 각 계층간 데이터 변환 없이 사용할 수 있어 수평적 통합이 아닌 수직적 통합이 가능하게 된다. 또한, 가장 두드러지는 이점은 이더넷 기술과 함께 사용할 수 있다는 것이다. 현재의 제어시스템은 산업용 이더넷의 발달로 PC 기반 제어시스템의 도입과 웹 기반 어플리케이션 및 IT기술의 발달, 모바일 통신 등 다양한 기술이 접목되어 발전하고 있다. 산업용 이더넷의 기술은 비용절감이라는 강점에 힘입어 제어기 간 통신은 물론 리모트 입출력장치 수준까지 통신을 이루고 있다. 불과 몇 년전에 시도된 노력이 급속도로 발전하고 있는 것이다. 더 나아가 앞으로는 필드 기기에 웹 서버를 내장하여 필드 기기의 데이터를 네트워크 계층의 최상위 존재하는 비즈니스 레벨까지 직접적으로 전달할 수 있음은 물론, 웹 기반 어플리케이션에서 전체 제어시스템의 통합과 관리가 이루어질 수 있을 것이다. 따라서 본 연구에서는 XDD 요소의 Variable 요소와 Menu 요소, PROFIBUS 요소의 GeneralKey 요소만을 이용하여 필드기기 표현 가능성을 설명하였으나 앞으로 더욱 발전시켜 사용자가 진정으로 바라는 개방형 제어시스템을 구성할 수 있도록 해야 할 것으로 생각된다.

참고문헌

[1] G. Kaplan, "Ethernet's winning ways," *IEEE Spectrum,* pp. 113-115, 2001.

[2] J. McGgilvreay, "The ethernet decision" Industrial Computing Online, 2000.

[3] J. Strothman, "Will web services replace HMI?," InTech, pp. 34-36, 2002.

[4] T. Tommila, O. Ventä, K. Koskinen, "Next generation industrial-need and opportunities," Automation Technology Review, pp. 34-41, 2001.

[5] P. Pinceti, "How will XML impact industrial automation?," www.isa.org, pp. 37-40, 2002.

[6] M. Wollschlaeger, "A framework for fieldbus management using XML description," WFCS-2000, September, 2000.

[7] D. Buhler, W. Kuchlin, "Remote fieldbus system management with java and XML," ISIE'2000, Cholula, Puebla, Mexico, 2000.

[8] D. Buhler, "The CANopen markup language -representing fieldbus data with XML," *IEEE,* pp. 2449-2454, 2000.

[9] P. Neumann, C. Diedrich, R. Simon, "Engineering of field devices using device description," IFAC. 15th Triennial World Congress, Barcelona, Spain, 2002.

[10] C. Diedrich, P. Neumann, "Field device integration in DCS engineering using a device model," *IEEE,* pp. 164-168, 1998.

[11] D. Hartenstein, "Using XML in distributed real time automation processes in the textile industry," Internationales Comgress Centrum, Berlin, Germany, 2001.

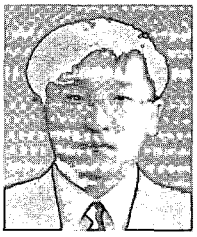
[12] J. Bono, "XML reaches factory floor's automation Islands"

[13] J. Hoch, "The ABCs of XML," Putman Media, 2000.

[14] G. Y. Tian, Z. X. Zhao, R. W. Baines, "A fieldbus -based intelligent sensor," *Mechatronics,* vol. 10, no. 835-49, 2000.

[15] A. V. Scott, W. J. Buchanan, "Truly distributed control systems using fieldbus technology," *IEEE,* pp. 165-173, 2000.

[16] L. Kim, "XML SPY - XML integrated development environments," *Altova, Inc.*, The XML Spy company, 2002.

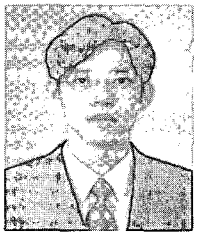


문 용 선

1983년 조선대학교 전자공학과 졸업. 1985년 동 대학원 석사. 1989년 동 대학원 박사. 1992년~현재 순천대학교 전자공학과 교수. 관심분야는 Vision-based Robot Control.

정 철 호

1982년 서울산업대학교 전자공학과 졸업. 2001년~현재 순천대학교 전자공학과 박사과정. 1987년~현재 (주)포스콘 근무. 관심분야는 산업용 실시간 네트워크 및 개방화 제어시스템 구현.



이 명 복

2002년 순천대학교 전자공학과 졸업. 2002년~현재 순천대학교 석사과정. 관심분야는 산업용 실시간 네트워크 및 개방화 제어시스템 구현.