15th Triennial World Congress, Barcelona, Spain

ENGINEERING OF FIELD DEVICES USING DEVICE DESCRIPTIONS

Peter Neumann, Christian Diedrich, René Simon

Institut für Automation und Kommunikation e.V. (ifak)
Steinfeldstraße 3 (IGZ), D-39179 Barleben, Germany
Internet: http://www.ifak.fhg.de, Email: rsi@ifak.fhg.de

Abstract: Device Descriptions are necessary for the integration of intelligent field devices in commissioning & maintenance tools, engineering systems or MES / ERP systems. Device Descriptions comprise of device models and presentations based on the models (e.g. ASCII files). The XML concept may help to enlarge the application scope of Device Description because it is becoming one of the basic technologies of the fast growing internet and various e-engineering activities. These issues are addressed from a modelling and implementation point of view. *Copyright © 2002 IFAC*

Keywords: Distributed Computer Control Systems, Formal Methods, Computer-aided Engineering

1. INTRODUCTION

Fieldbus systems are a part of most modern complex machines and plants. They provide unique methods for signal interchange between control systems and field devices at the communication layer.

A large number of field devices of different vendors are used in complex systems. The device vendors supply their proprietary tools together with field devices. The handling of these tools differs between vendors. In this way, many different tools must be handled in all life cycle phases of the plant. Data must be exchanged between these tools. This data exchange is not standardised, therefore data conversions are often necessary, requiring detailed specialist knowledge. In the end, the consistency of data, documentation and configurations can only be guaranteed by an intensive system test. Thus, the large number of different device types and suppliers within a control system project makes the configuration task difficult and time-consuming (Neumann, et al., 1995).

The central workplace for service and diagnostic tasks in the control system does not fully cover the functional capabilities of fieldbus devices. Furthermore, the different device-specific tools cannot be integrated into the system's software tools. Typically, device-specific tools can only be connected directly to a fieldbus line or directly to the field device.

In order to maintain the continuity and operational reliability of process control technology, it is necessary to fully integrate fieldbus devices as a sub-component of process automation (ESPRIT Project 6188, 1995). The following sections introduce approaches designed to eliminate the insufficiencies described above.

The analysis is done investigating the overall engineering process of the DCS life cycle and determining, in which activities field devices are used (instrumentation). On this basis, a field device model is developed (using UML technologies as class diagrams). This device model can be implemented (realised, used) in several ways, here the Electronic Device Description (EDD) and a XML approach are shown. EDD is already an industrial standard, whereas XML based approaches are emerging now, but have to re-use the knowledge already developed in the automation area.

2. ENGINERING & INSTRUMENTATION

Besides the functional view on control systems, the life cycles of these systems (engineering) is becoming more and more important (Simon, R., Hörger, J., 1999). According to (Alznauer, 1998) engineering with regard to the control system is understood as "… all tasks and activities which are carried out for the planning, construction, commissioning, operation and maintenance of technical systems".

The life cycle and engineering are becoming more complex as the used devices and tools are becoming more differentiated and complex.

The non-interrupted engineering on the basis of common information models and formalised description techniques is a new quality in control systems and a basic requirement for the design of new generation control systems. These control systems are called Distributed Control Systems – DCS here.

The handling of the life cycle of DCSs is a complex process which can only be done using sophisticated tools (hardware, software). Here it is very important to design a non-interrupted the life cycle, i.e. to achieve a information transfer from one step to another step without losing information, and to ensure a single-source principle while putting information into the system. This information is used in each step to create a special view for the user. This can not be done using paper documents for storing and transporting information. However, the usage of data base management systems does not as long as open technologies are not applied. It is necessary to use databases and communication systems which follow a commonly agreed transfer syntax and standardised information models which ensure the meaning of the information (semantics). Basically, a connection between all tools in a DCS must be created (Diedrich, Ch., Neumann, P., 1998b).

There are several possibilities to classify the life cycle / the engineering of DCS (Alznauer, 1998). Such classification can be done using:

- the hierarchy of the control functions
- their timing sequence
- their logical dependencies

All life cycle phases which are connected to field devices should be united under the word "instrumentation" and described as use cases.

Instrumentation is defined as follows:

> Instrumentation comprises of all activities within the life cycle of the Distributed Control System where handling of the field devices (logically or physically) is necessary.

Therefore, instrumentation can be considered as the intersection of the life cycle of the Distributed Control System and the field device.

The following figure shows the instrumentation steps as an UML use case diagram. There is only one actor, who is not described in detail.

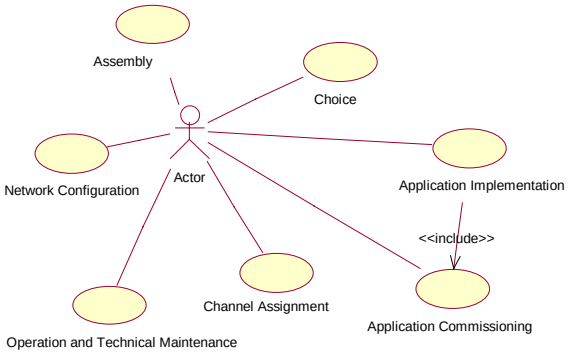

Fig. 1. Use case Diagram Instrumentation

3. DEVICE MODEL

Field devices are linked both with process, via input /output hardware/ software, and with other devices via communication controllers/ transmission media. The centre of our attention is on the field device as the computational power is increasing rapidly as mentioned above. Thus, the applications are run more and more on these devices and the application processes are becoming more and more distributed. We have to solve the problem of configuring and parameterisation of these field devices during the operation for real-time data processing purposes, diagnosis, parameter tuning etc. Therefore, there is a need to model such field devices (Diedrich, Ch., Neumann, P., 1998a).

A field device can be characterised by:

- internal data management (process I/O image, communication parameters, application parameters)
- process interface
- information processing (e. g. Function Blocks)
- communication interface (Fieldbus, Ethernet-TCP/IP etc.)
- (optional) man/ machine interface (local display, buttons, switches, LEDs)
- (optional) persistent memory and others.

We can define a device model as shown in Figure 2 (represented by UML packages) which supports the data exchange between instrumentation steps. This is a very abstract presentation of a device model.

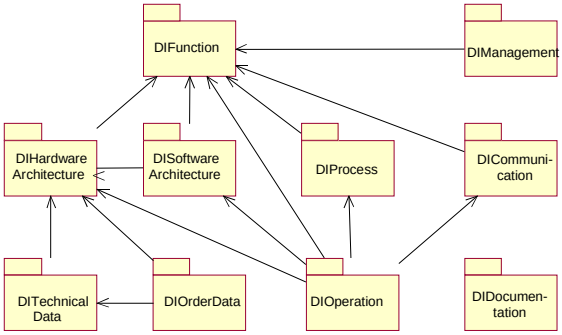

Fig. 2. Device model related to the instrumentation

Important are the packages DIFunction, DIHardwareArchitecture, DISoftwareArchitecture, DIProcess, DICommunication, DIManagement and DIOperation (DI stands for Device Instrumentation). The packages depicted in Figure 2 contain the detailed model represented by UML class diagrams modelling the different views on a device.

Figure 3 depicts the class diagram of the package DIFunction.

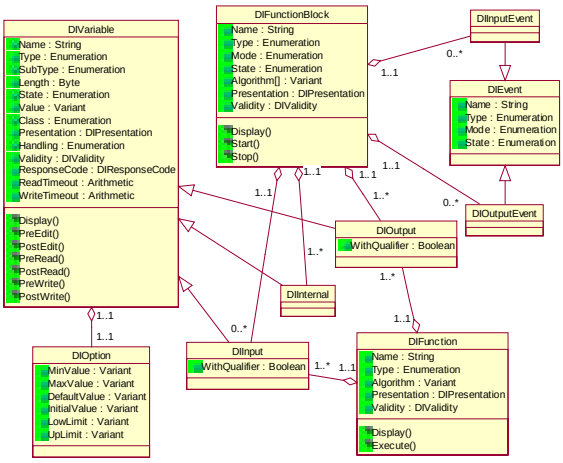

Fig. 3. Package DIFunction

Figure 4 depicts the class diagram of the package DICommunication.

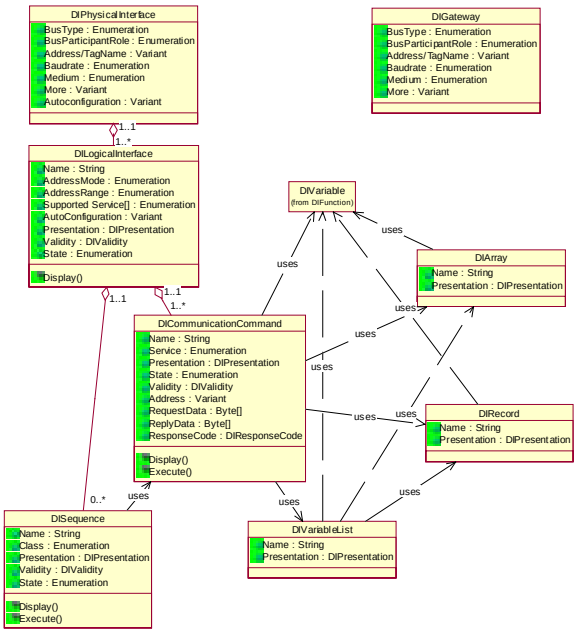

Fig. 4. Package DICommunication

Figure 5 depicts the class diagram of the package DIOperation.

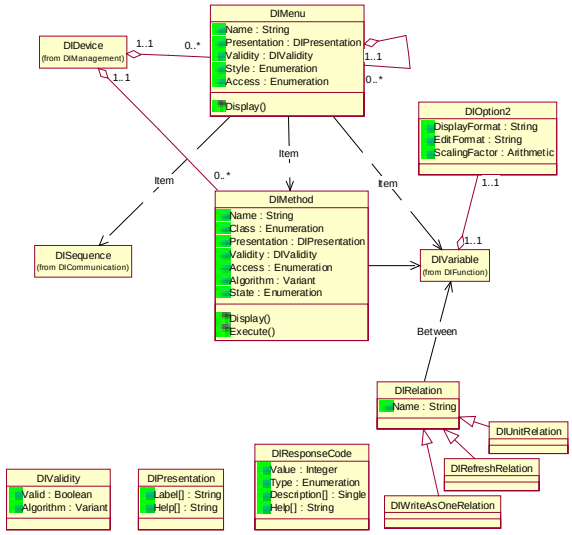

Fig. 5. Package DIOperation

(Simon, R., 2001) contains all other class diagrams needed for modelling the semantics of field devices as well as further explanations. This model has to be described by description languages to generate the basic information for an uninterrupted tool chain.

4. DESCRIPTION AND REALISATION OPPORTUNITIES

4.1 Electronic Device Description (EDD)

The device model can be implemented (realised) in several ways. For the computable description of device parameters for automation systems components, the so called Electronic Device Description Language (EDDL) has been specified (NOAH, 1999), (PNO, 2000), (Simon, R. Demartini, C.,

1999b). EDD is used to describe the configuration and operational behaviour of a device and covers the following aspects (Neumann, et al., 2001):

- description of the device parameters, semantically defined by the field device model mentioned above
- support of parameter dependencies
- logical grouping of the device parameters
- selection and execution of supported device functions
- description of the device parameter access method.

The "C-based" EDDL describes product data in a declarative way mixed with some programming features. It includes description elements to address blocks and single parameters, capitalised words are part of the EDDL syntax specification.

Parameters (VARIABLE) are identified by their name. All other parts describing the behaviour of the device and the relations between device parameters refer to these names. Each parameter is defined by its data type (TYPE) and a read/write access handling (HANDLING).

Some runtime conditions may lead to a situation where a parameter is not valid. This feature is expressed with a VALIDITY identifier. The representation of the parameter at the user interface is controlled by LABEL and HELP strings within the parameter declaration paragraph.

A communication action may be initiated by calling a read or write COMMAND as identified by a name string. Within this declaration, parameters are referenced to be transferred. The operation of a command determines whether a variable is read or written from host to device. Furthermore, the absolute or relative addressing scheme is specified by the COMMAND structure.

MENU constructs are used to organise parameters, methods and other items specified in the EDDL into a hierarchical structure. A host application may use the menu items to display information to the user in an organised and consistent fashion. METHODs are the most programming language like features of the EDDL. They describe the execution of interactions that occur between host devices and a field device, or between field device parameters and local parameters. The method declarations use C-like syntax. It is possible to access device parameters via read and write commands which refer to the related COMMAND declarations. Furthermore, the EDD interpreter serves some built-in functions which are accessed by name strings reserved by the EDDL syntax specification (e.g. selection dialogue functions etc.). EDDL supports further extended features such as complex data types, file import, creating similar items with LIKE constructs and others. EDDL is not a full featured programming language. General features such as memory management and system access are not necessary.

The EDD can be considered as a configuration related electronic data sheet for Field Devices (especially for PROFIBUS devices). It can be delivered either on disc, bundled with the device, or via the internet. It can even reside in every device. The EDD can be accessed either from the configuration tool repository representing the collection of EDDs or directly from the device if the EDD resides in this device. Thus, the consistency of the device version and its associated EDD can easily be checked.

Devices of different vendors solving similar tasks usually possess many parameters with equal semantics. The EDDL offers the possibility to use standard libraries which contain standard blocks or item collections. The parameters in a library may be completely described with user presentation information and related communication commands. These libraries may be imported by various EDDs. Properties of a device which differ from the library specifications may be easily redefined. Thus, different devices get uniform presentations for the used library items.

4.2 Overall Example

EDD is based on the ASCII standard. XML (Extended Markup Language) could be a promising approach for the future, especially because of its use in other areas. Both approaches contain definitions for the exchange of device descriptions using files. These definitions are not given here, however a small example is used to show that different realisations can and must be based on the same solid foundation - the device model.

The example comprises of a variable (package DIFunction), which is described by name, data type, label and help.

The class DIVariableExample as shown in the following figure is the starting point.

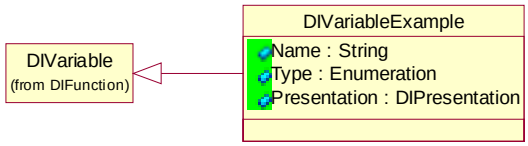

Fig. 6. Class DIVariableExample

The realisation as Electronic Device Description Language is described using language production rules shown in the following figure.

```
variable
        =       'VARIABLE'      Identifier      '{'
variable_attribute_list '}'

variable_attribute_list
        = variable_attribute_listR

variable_attribute_listR
        = variable_attribute
```

```
        =                   variable_attribute_listR
variable_attribute

variable_attribute
        = help
        = label
        = type
```

Fig. 7. Language production rules (EDD)

A sentence created according to these rules may look like the following figure.

```
VARIABLE   Temperature
{
  LABEL       "Temperatur";
  TYPE        DOUBLE;
  HELP        "Temperatur";
}
```

Fig. 6. A variable definition (EDD)

Using this definition of a variable, a commissioning tool provides the following human machine interface:

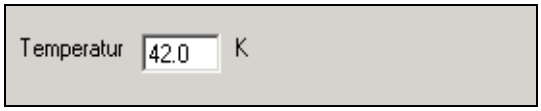

Fig. 8. Human machine interface showing a variable

The unit (Kelvin) is not supported by the example model. The help text is not visible, the name of the variable not used. Based on the data type, the value provided by the device is shown.

4.3 The XML approach

The eXtensible Markup Language XML (Bray, et al., 1998) expands the description language HTML with user-defined tags, data types and structures. In addition, a clear separation between the data descriptions, the data themselves, and their representation in a browser has been introduced. Furthermore, declaring syntactical and semantical information in a separate file (Document Type Definition, DTD), allows re-using the description structure in different contexts. This provides a number of benefits when using the same XML description file for different tasks. Different views can be implemented on top of the same data. The description can be hierarchically organised. Depending on the functions to be performed, the XML data can be filtered and associated to software components (controls, Java beans, etc.). The selection of the necessary information and the definition of their presentation details can be performed by means of scripts and style sheets. The style sheets are part of the development of XML (Boumphrey, 1998). In most cases they are implemented using the extensible Style Language (XSL). The XML file, the scripts and the different style sheets can be used to generate HTML pages, special text files, and binary files (components, applets) necessary to build the certain functions of the software tools. The distribution of the generated HTML pages and associated software components is done following the concepts used in an Internet environment. The major benefit of this solution is a unique, reusable description with an excellent consistency and reduced efforts of the description process.

For the realisation using the XML approach, the specification of a schema is necessary. The following figure shows part of it describing the element variable.

```
    <ElementType  name="DIVariable"  content="mixed"
model="closed">
          <attribute type="Name" required="yes"/>
          <element  type="operation:DIPresentation"
minOccurs="1" maxOccurs="1"/>
          <element    type="Type"    minOccurs="1"
maxOccurs="1"/>
    </ElementType>
```

Fig. 9. Schema specification

An instance of this schema may look like the following:

```
    <DIVariable
          Name="Temperature"
          <operation:DIPresentation>
                <operation:Label
                      String="Temperatur">
                </operation:Label>
                <operation:Help
                      String="Temperatur">
                </operation:Help>
          </operation:DIPresentation>
          <Type>
                <Arithmetic>
                      <Integer
                      </Integer>
                </Arithmetic>
          </Type>
    </DIVariable>
```

Fig. 10. A variable definition (XML)

A standard web browser creates the following interface:

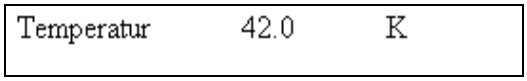

Fig. 11. Web browser showing a variable

The unit (Kelvin) is not supported by the example model. The help text is not visible, the name of the variable not used. Based on the data type, the value provided by the device is shown.

The presentation of the small example underlines the objectives targeted by the modelling approach. If the internal structures of different realisations are similar, i.e. follow the same field device model, then it is possible to build translators from one realisation to another and to secure investments already done. Similar presentations with the same contents provide the opportunity to simplify training and education through previous recognition.

5. SUMMARY

Device Descriptions are necessary for the integration of intelligent field devices in commissioning tools, maintenance tools, engineering systems or MES / ERP systems. Device Descriptions comprise of device models and presentations based on the models (e.g. ASCII files). The XML concept may help to enlarge the application scope of Device Description because it is becoming one of the basic technologies of the fast growing internet and various e-engineering activities. At present, we can observe a transition to XML based Device Descriptions which is characterised by the following issues:

- XML and ASCII based Device Description have to use the same device model to keep the semantics already developed by automation industry
- new application functions supporting different phases of the life cycle of an automation system have to be defined using different views on the Device Description.
- consequently, a new chain of tools for creating and using Device Descriptions has to be developed.

There are a lot of other realisations which can profit from a common device model:

- the firmware of the device itself
- type & instance manager in software tools
- component interfaces (e.g. (PNO, 2000a))
- device & application profiles (e.g. (PNO, 1997))
- proxy / function block within a PLC
- e-business solutions

A necessary precondition is the international standardisation in this area, which is currently done in the European standardisation CENELEC and the IEC.

REFERENCES

Alznauer, R. (1998): Semantic information model for a overall, computer based engineering using the process industry as example(in German). PhD thesis, Rheinisch Westfälische Technische Hochschule Aachen.

Boumphrey, F. (1998). Professional Style Sheets for HTML and XML. Wrox Press.

Bray, T., Paoli, J., Sperberg-McQueen, C. M. (1998). Extensible Markup Language (XML) 1.0., http://www.w3.org/TR/REC-xml.

Diedrich, Ch., Neumann, P. (1998a). Field device integration in DCS engineering using a device model, IECON'98, IEEE Conference, Proceedings pp. 164-168, Aachen.

Diedrich, Ch., Neumann, P. (1998b). Standardisation in Automation Systems, Proc. of the 9th IFAC Symposium on Information Control in Manufacturing (INCOM´98) pp. 1:51-56 Nancy, 1998.

ESPRIT Project 6188 (1995). Prenormative Requirements for Intelligent Actuation and Measurement, Deliverable D12.2&3, PRIAM.

Neumann, P., Diedrich, Ch., Simon, R. (1995). Necessary extensions of fieldbus systems for distributed processing, Proceedings IEEE International Workshop on Factory Communication Systems WFCS'95, Proceedings pp. 247-254.

Neumann, P., Simon R., Diedrich, Ch.,, Riedl, M. (2001). Field Device Integration. 8th IEEE International Conference on Emerging Technologies and Factory Automation. ETFA 2001, Proceedings pp.63-68, Antibes.

NOAH (1999). Language Specification of Electronic Device Description, Deliverable 321, Network Oriented Application Harmonisation (NOAH).

PNO (1997). PROFIBUS-PA Profile for Process Control Devices, V2.0, PROFIBUS User Organisation.

PNO (2000a). Field Device Tool Interface Specification, Version 1.0, PROFIBUS Guideline. PROFIBUS User Organisation.

PNO (2000b). Electronic Device Description, Version 1.0. PROFIBUS Guideline. PROFIBUS User Organisation.

Simon, R., Hörger, J. (1999a). Engineering of Distributed Automation Systems Based on Novel Information Technologies and Methods, Fet'99, 23./24.09.1999, Magdeburg, Proc. of FET'99, pp.223-229, ISBN 3-211-83394-3, Springer Verlag Wien New York.

Simon R., Demartini, C. (1999b). Electronic Device Description, Fet'99, 23./24.09.1999, Magdeburg, Proc. of FET'99, pp.429-436, ISBN 3-211-83394-3, Springer Verlag Wien New York.

Simon, R. (2001). Methods for Field Instrumentation of Distributed Computer Control Systems (in German). PhD Thesis, Otto-von-Guericke University Magdeburg.