65/314A/NP

NEW WORK ITEM PROPOSAL

	Proposer Secretariat	Date of proposal 2003-04
	TC/SC TC 65	Secretariat France
Classification according to IEC Directives Supplement, Table 1	Date of circulation 2003-05-09	Closing date for voting 2003-08-08

A proposal for a new work item within the scope of an existing technical committee or subcommittee shall be submitted to the Central Office. The proposal will be distributed to the P-members of the technical committee or subcommittee for voting, and to the O-members for information. The proposer may be a National Committee of the IEC, the secretariat itself, another technical committee or subcommittee, an organization in liaison, the Standardization Management Board or one of the advisory committees, or the General Secretary. Guidelines for proposing and justifying a new work item are given in ISO/IEC Directives, Part 1, Annex C (see extract overleaf). **This form is not to be used for amendments or revisions to existing publications.**

The proposal (to be completed by the proposer)

Title of proposal Device profile guideline		
☐ Standard	☐ Technical Specification	☒ Publicly Available Specification
Scope (as defined in ISO/IEC Directives, Part 2, 6.2.1) See page 7		
Purpose and justification, including the market relevance and relationship to Safety (Guide 104), EMC (Guide 107), Environmental aspects (Guide 109) and Quality assurance (Guide 102) . (attach a separate page as annex, if necessary)		
Target date	for first CD	for IS .2003-10
Estimated number of meetings	Frequency of meetings: per year	Date and place of first meeting: ....
Proposed working methods	☐ E-mail	☐ ftp
Relevant documents to be considered		
Relationship of project to activities of other international bodies		
Liaison organizations	**Need for coordination within ISO or IEC** SC17B, SC22G, TC 57, SC65A, SC65B, SC65C, ISO TC184 SC5	
Preparatory work Ensure that all copyright issues are identified. Check one of the two following boxes ☐ A draft is attached for vote and comment ☐ An outline is attached We nominate a project leader as follows in accordance with ISO/IEC Directives, Part 1, 2.3.4 (name, address, fax and e-mail):		
Concerns known patented items (see ISO/IEC Directives, Part 2)	**Name and/or signature of the proposer**	

☐ yes If yes, provide full information as an annex ☐ no	

Comments and recommendations from the TC/SC officers
1) Work allocation ☐ Project team ☐ New working group ☐ Existing working group no:
2) Draft suitable for direct submission as ☐ CD ☐ CDV ☐ Publication as a PAS
3) General quality of the draft (conformity to ISO/IEC Directives, Part 2) ☐ Little redrafting needed ☐ Substantial redrafting needed ☐ no draft (outline only)
4) Relationship with other activities In IEC In other organizations
Remarks from the TC/SC officers This document is based on 65/290/DC and takes into account the comments made by National Committees on 65/290/DC. The resolution of comments made by the PJWG is available in the revised compilation of comments 65/296A/INF. Following the resolution #4 taken during TC65 Beijing meeting (see 65/294/RM) this document is circulated as an internal PAS

Elements to be clarified when proposing a new work item

Title

Indicate the subject matter of the proposed new standard.

Indicate whether it is intended to prepare a standard, a technical report or an amendment to an existing standard.

Scope

Give a clear indication of the coverage of the proposed new work item and, if necessary for clarity, exclusions.

Indicate whether the subject proposed relates to one or more of the fields of safety, EMC, the environment or quality assurance.

Purpose and justification

Give details based on a critical study of the following elements wherever practicable.

a) The specific aims and reason for the standardization activity, with particular emphasis on the aspects of standardization to be covered, the problems it is expected to solve or the difficulties it is intended to overcome.

b) The main interests that might benefit from or be affected by the activity, such as industry, consumers, trade, governments, distributors.

c) Feasibility of the activity: Are there factors that could hinder the successful establishment or general application of the standard?

d) Timeliness of the standard to be produced: Is the technology reasonably stabilized? If not, how much time is likely to be available before advances in technology may render the proposed standard outdated? Is the proposed standard required as a basis for the future development of the technology in question?

e) Urgency of the activity, considering the needs of the market (industry, consumers, trade, governments etc.) as well as other fields or organizations. Indicate target date and, when a series of standards is proposed, suggest priorities.

f) The benefits to be gained by the implementation of the proposed standard; alternatively, the loss or disadvantage(s) if no standard is established within a reasonable time. Data such as product volume of value of trade should be included and quantified.

g) If the standardization activity is, or is likely to be, the subject of regulations or to require the harmonization of existing regulations, this should be indicated.

If a series of new work items is proposed, the purpose and justification of which is common, a common proposal may be drafted including all elements to be clarified and enumerating the titles and scopes of each individual item.

Relevant documents

List any known relevant documents (such as standards and regulations), regardless of their source. When the proposer considers that an existing well-established document may be acceptable as a standard (with or without amendments), indicate this with appropriate justification and attach a copy to the proposal.

Cooperation and liaison

List relevant organizations or bodies with which cooperation and liaison should exist.

Preparatory work

Indicate the name of the project leader nominated by the proposer.

DEVICE PROFILE GUIDELINE

CONTENTS

FIGURES

INTRODUCTION

This guideline is a recommended outline for use by standardisation product committees, fieldbus consortia and product manufacturers to develop and provide profiles for networked devices. Some aspects of this guideline may also be applicable to stand-alone devices. The present wide variation in the form of concepts and methods used for disclosing device information and behaviour to users of devices leads to longer evaluations required to understand how to use and apply networked industrial devices. This variation makes determining device interoperability, interchangeability, comparisons and common device behaviour more difficult. Therefore it is the intention of this guideline to provide a common and more generic way to publish device information and behaviour. This is a contribution to reduce the total cost of the industrial control system.

Profiles define a common set of functionality for a class of devices in a given industrial domain, thus allowing system designers, system integrators and maintenance staff to handle profile-based devices without special tool configuration. They also allow consistent structuring and semantics of device functionality.

NOTE Other technologies are available to support the integration of devices into control systems, in particular to handle manufacturer-specific extensions in commissioning and engineering tools. Examples of such technologies are device description languages, which detail the internal structure of the device, or standardized software interfaces, where each device is represented by a dedicated software component.

Figure 1 shows the various possible profile documents and the typical writer of each document. The figure also illustrates the developing sequence for the developing of the profile documents. It is proposed that this profile guideline is the base for other working groups to develop profile standards and product class profiles. The root device profiles and the manufacturer device profiles can be developed from these profile standards. Finally the manufacturer can create the specific device descriptions for his products.

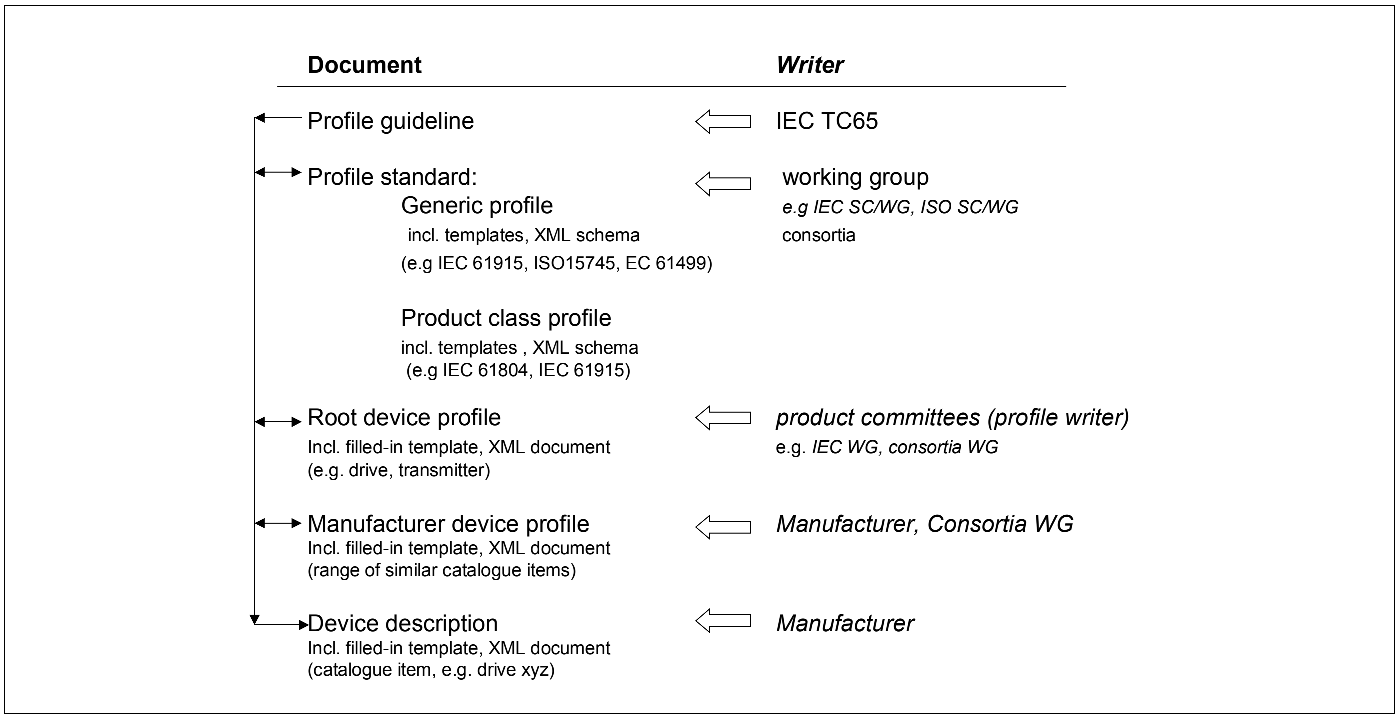

Document
Writer
Profile guideline
IEC TC65
Profile standard:
Generic profile
incl. templates, XML schema
(e.g IEC 61915, ISO15745, EC 61499)
working group
e.g IEC SC/WG, ISO SC/WG
consortia
Product class profile
incl. templates , XML schema
(e.g IEC 61804, IEC 61915)
Root device profile
Incl. filled-in template, XML document
(e.g. drive, transmitter)
product committees (profile writer)
e.g. IEC WG, consortia WG
Manufacturer device profile
Incl. filled-in template, XML document
(range of similar catalogue items)
Manufacturer, Consortia WG
Device description
Incl. filled-in template, XML document
(catalogue item, e.g. drive xyz)
Manufacturer

Figure 1 – Profile documents and their profile writer

This guideline provides the context, recommended minimum contents and construction rules of device profiles. Recommended generic device models, appropriate analysis and design diagrams using standards as UML (Unified Modelling Language) and methods to construct those models are provided.

This guideline provides recommendations for conveying the necessary device information to non-human users of the device profile such as software tools and application programs in an electronic file. These recommendations include the use of standards such as XML (eXtensible Markup Language).

1 Scope

This document provides guidance for the development of device profiles for industrial field devices and control devices, independent of their complexity.

NOTE 1 Examples of devices covered are limit switches and contactors for simple bit level device networks, medium complex devices such as transmitters and actuators for process control and complex device for field buses such as variable speed drives.

NOTE 2 This guideline is also recommended to be used for devices such as programmable controllers, network components and HMI. If a device is user programmable, its features, as introduced in this guideline (e.g. parameters and behaviour), cannot be completely described in the profile. However profile writers may agree on general common functions like Start, Stop and Reset as well as identification and process inputs/outputs.

A device profile covers physical, functional, communication and application aspects, irrespective of whether these aspects are accessible over the network.

NOTE 3 Different users of a device profile such as device manufacturers, system integrators and maintenance operators may only use specific aspects of the profile.

The guideline is written in an network independent way. Therefore it is applicable for various fieldbusses. The guideline is intended to be used by IEC product standards committees and industrial communications networks consortia when they develop their device profile organizations and structures. It is not intended to provide an outline for a specific device profile. Further this guideline presents device models to better guide and delineates a device profile’s content. The profile guideline allows the use of a parameter list, function block model and/or object model to convey the structure and behaviour of the device in a unique manner. It is up to the profile writers to decide which of the models they apply.

To be useful to users a common method for conveying the device profile information is required. This guideline recommends the use of device profile templates. This guideline gives an example of a template which is intended to be the basis of the structure and content of further templates which may be developed by the relevant profile groups.

This will allow users of these profiles to make comparisons, determine interoperability and interchangeability, and recognize common device behaviour.

The development of industrial application and process profiles as covered by the ISO 15745-1 is not within the scope of this guideline.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61131-3:2003, *Programmable controllers – Part 3: Programming languages*

IEC/PAS 61499-1:2000, *Function blocks for industrial process measurement and control – Part 1: Architecture*

IEC/PAS 61499-2:2001, *Function blocks for industrial process measurement and control – Part 2: Software tool requirements*

IEC/PAS 61804-1:2002, *Function blocks (FB) for process control – Part 1: Overview of system aspects*

IEC/PAS 61804-2:2002, *Function blocks (FB) for process control – Part 2: Specification of FB concept and Electronic Device Description Language (EDDL)*

IEC/TS 61915:2003, *Low-voltage switchgear and controlgear – Principles for the development of device profiles for networked industrial devices*

ISO/IEC DIS 19501-1:—, *Information Technology – Unified Modelling Language (UML) – Part 1: Specification.*

ISO 15745-1:2003, *Industrial automation systems and integration – Open systems application integration framework – Part 1: Generic reference description*

3 Definitions and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1 algorithm

completely determined finite sequence of operations by which the values of the output data can be calculated from the values of the input data [IEC Dictionary 351-11-21]

3.1.2 application

software *functional element* specific to the solution of a problem in industrial-process measurement and control

NOTE An application may be distributed among *resources*, and may communicate with other applications.

3.1.3 attribute

property or characteristic of an entity

3.1.4 class

description of a set of *objects* that share the same *attributes*, *operations*, methods, relationships, and semantics [OMG]

3.1.5 data

reinterpretable representation of *information* in a formalized manner suitable for communication, interpretation or processing. [ISO 2382, 01.01.02]

3.1.6 data type

set of values together with a set of permitted *operations* [ISO 2382, 15.04.01]

3.1.7 device

field device

networked independent physical *entity* of an industrial automation system capable of performing specified functions in a particular context and delimited by its *interfaces* [IEC 61499-1]

entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system [ISO 15745-1]

3.1.8 device class

set of devices with a defined functional commonality in terms of their *parameters* or *functional elements*

3.1.9 **device profile**

representation of a device in terms of its parameters, parameter assemblies and behaviour according to a device model that describes the device’s data and behaviour as viewed through a network

NOTE This is a definition from IEC/TS 61915 which is extended by the addition of the device functional structure.

3.1.10 entity

particular thing, such as a person, place, *process*, object, concept, association, or *event* [dpANS X3.172, 1989]

3.1.11 execution

process of carrying out a sequence of *operations* specified by an *algorithm* [ISO]

3.1.12 functional element

entity of software or software combined with hardware, capable of accomplishing a specified function of a *device*

NOTE 1 A functional element has an interface, associations to other *functional elements* and functions

NOTE 2 A functional element can be made out of function block(s), object(s) or parameter list(s)

3.1.13 function block

software functional element comprising an individual, named copy of a data structure and associated *operations* specified by a corresponding *function block type*

NOTE Adapted from IEC 61499

3.1.14 input data

data transferred from an external source into a *device*, *resource* or *functional element*

3.1.15 instance

functional element comprising an individual, named copy of a data structure and associated *operations* specified by a corresponding *functional element type.*

3.1.16 interface

shared boundary between two *entities* defined by functional characteristics, signal characteristics, or other characteristics as appropriate [IEC 60050-351]

NOTE The interface typically includes the device parameters.

3.1.17 method

implementation of an operation, which specifies the algorithm or procedure associated with an operation.

3.1.18 model

mathematical or physical representation of a system or a process, based with sufficient precision upon known laws, identification or specified suppositions [IEC Dict 351-11-20]

3.1.19 object

entity with a well-defined boundary and identity that encapsulates state and behaviour [UML]

NOTE State is represented by attributes and relationships; behaviour is represented by operations, methods, and state machines. An object is an instance of a class.

3.1.20 operation

service that can be requested from an object to effect behaviour [UML]

3.1.21 output data

data originating in a device, resource or functional element and transferred from them to external systems.

3.1.22 parameter

specification of a variable that can be changed, passed, or returned

NOTE A parameter may be characterised by a name, type and direction.

3.1.23 resource

- logical device
- module

- group of functional elements which has independent control of its operation, and which provides various services to applications, including the scheduling and execution of algorithms.

NOTE The RESOURCE defined in IEC 61131-3 is a programming language element corresponding to the *resource* defined above.

3.1.24 service

specific work performed by a *device* or *object.*

3.1.25 type

hardware or software element which specifies the common *attributes* shared by all *instances* of the type.

3.1.26 use case

class specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system [OMG].

3.1.27 variable

software entity that may take different values, one at a time [ISO 2382]

NOTE The values of a variable as well as of a parameter are usually restricted to a certain data type.

Abbreviated terms

AIP – Application Interoperability Profile

DCS – Distributed Control System

ERP – Enterprise Resource Planning

FBD – function block Diagram

HMI – Human Machine Interface

H/W – Hardware

I/O – Input/Output

MES – Manufacturing Execution System

OMG –Object Management Group

S/W – Software

UML – Unified Modeling Languages

URL – Universal Resource Locator

XML – Extensible Markup Language

4 Guideline overview

The device profile guideline:

presents a short introduction into the entire scope of profiles

specifies the subset which is the focus of this guideline

introduces a general structural view to a device

A sequence of six profile definition steps is proposed to the profile writers groups to develop the necessary information for a device profile. This is recorded in a profile template which is introduced in an according clause. The profile template is to be collected in an electronically readable form and in a printed human readable document.

This guideline is based on the three typical approaches used in the automation industry,

the parameter list model,

the function block model and,

the object model.

The guideline recommends using one of these three models. As a minimum, the parameter list model should be applied to be in line with this guideline. Further models based on the parameter list model are possible provided that they may be mapped to one of the models.

Special annexes provide the model based background of the profile development steps and the template.

Several annexes provide additional information and material for the profile writer groups.

5 Automation Model and Device Profiles

5.1 ISO 15745 “Open systems application integration frameworks”

There are several aspects to be considered during device profiling. Figure 2 shows where the device profile fits in relation to other profiles necessary to build an automation application. Figure 3 shows a typical hardware implementation of an automation application. Figure 4 shows an example of a functional device structure.

According to ISO 15745 “Open systems application integration frameworks” a general application includes the technological process which has a material and energy flow, equipment and devices which carries these flows, devices which carries out the information processing, the communication systems connecting these devices as well as human beings interaction with the devices and at least the process. The ISO 15745-1 model contains each component of this system (see Figure 2). The automation devices are a subset of this entire framework which is in the scope of this profile guideline. Therefore this profile guideline deals with the device model and device profile parts of ISO 15745-1 only (the scope is highlighted in Figure 2 using an ellipse). The communication network integration model, profile and specifications are outside the scope of this guideline.

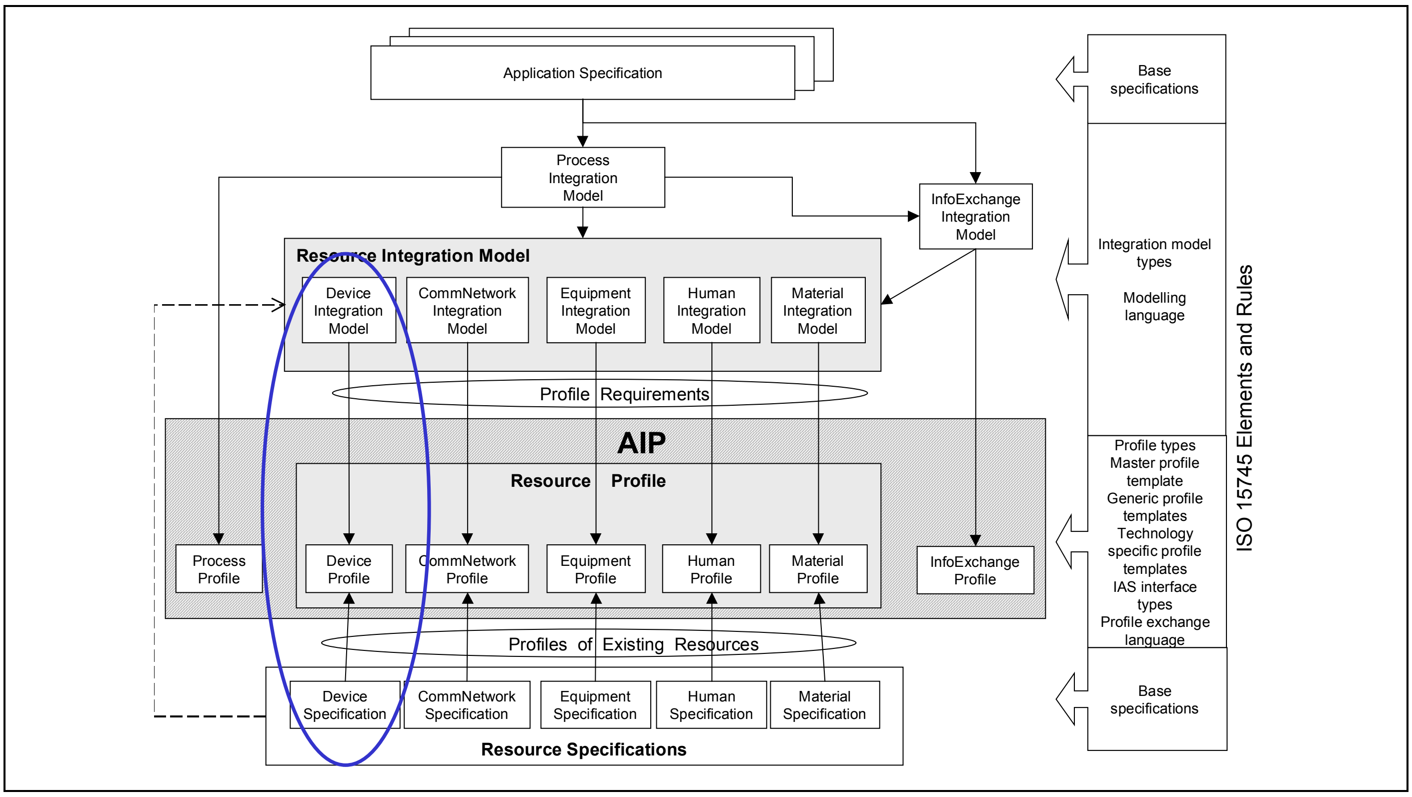


Figure 2 – Profile development using ISO 15745-1

Legend:

——> : indicates order in which activities are performed

- - - > : indicates information flow

5.2 Typical automation configuration

The field device is typically integrated as a component in an industrial automation system. The automation system performs the automation related part of the entire application. To define the complete profile of a specific field device class it is necessary for the profile writers to agree on the interactions of the device with the other components of the automation system: functions and corresponding required interfaces (including the defined device parameters). The components of an industrial automation system may be arranged in multiple hierarchical levels connected by communication systems as illustrated in Figure 3.

The field devices are components in the process level connected via inputs and outputs to the process or the physical or logical sub-networks. This also includes programmable devices and routers or gateways.

A communication system (e.g. a field bus) connects the field devices to the upper level controllers, which are typically programmable controllers or Distributed Control Systems (DCS) or even Manufacturing Execution Systems (MES). Since the engineering tools and the commissioning tools should have access as well to the field devices and as to the controllers, these tools are also located at the controller level. The “intelligent” field devices may directly communicate with each other via the fieldbus or the controller (Programmable Controller).

In larger automation systems another higher level may exist, connected via a communication system like LAN or Ethernet. In these higher-level visualization systems (HMI), DCS, central engineering tools and SCADA are located. Multiple clusters of field devices (with or without a controller, as described above) may be connected over the LAN with each other or to the higher-level systems.

Manufacturing Execution System (MES), Enterprise Resource Planning (ERP) and other Information Technology (IT) based systems can have access to field devices indirectly via the LAN and the controllers or directly via routers.

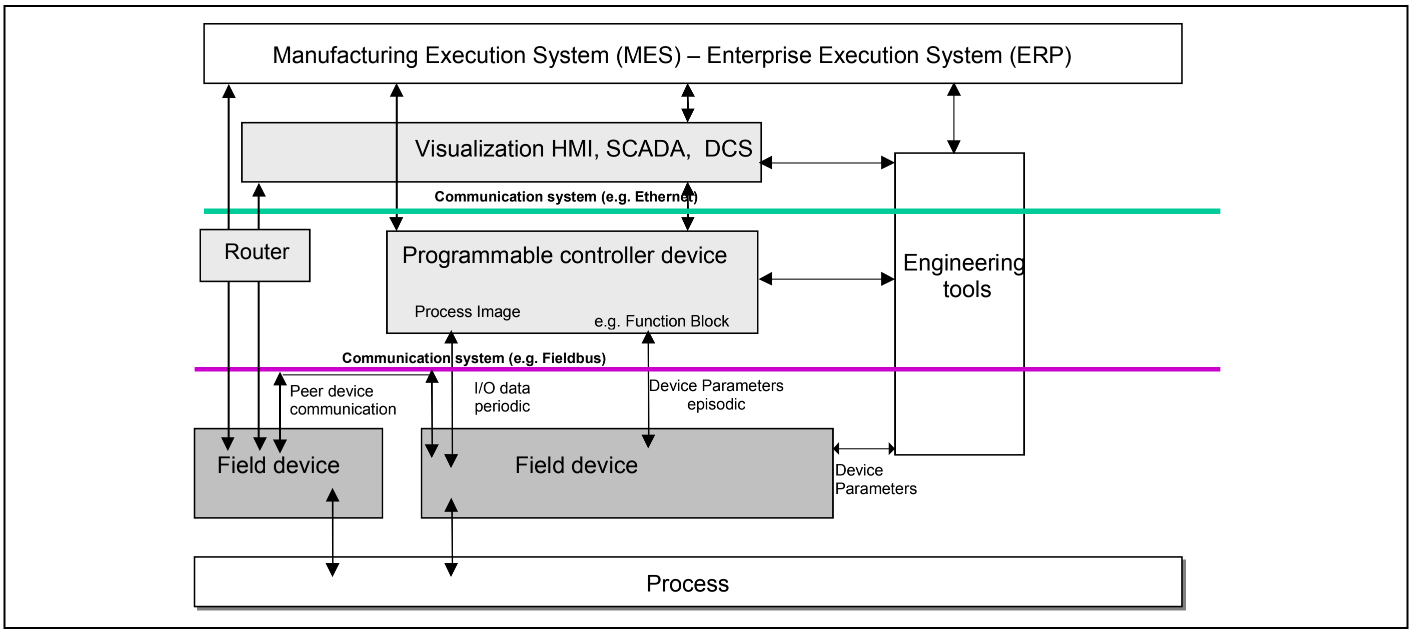

Manufacturing Execution System (MES) – Enterprise Execution System (ERP)
Visualization HMI, SCADA, DCS
Communication system (e.g. Ethernet)
Router
Programmable controller device
Process Image
e.g. Function Block
Engineering tools
Communication system (e.g. Fieldbus)
Peer device communication
I/O data periodic
Device Parameters episodic
Field device
Field device
Device Parameters
Process

Figure 3 – Typical automation configuration

NOTE Dark grey boxes represent field devices which are in the main scope of profiling. Light grey boxes are network connected devices which may also be considered as devices in the scope of the guideline.

5.3 Modular device structure

The device can be structured in a hierarchical way as shown in Figure 4. The main components of the structure can be containers which are known as modules (e.g. in the remote I/O domain), resources (e.g. in IEC 61131-3 and IEC 61499) or logical devices (e.g. within some fieldbusses), which can be further subdivided into *functional elements*. *Functional element* is a generic term for parameter lists members, function blocks and objects. All *functional elements* have parameters and optional behaviour. Modules, resources and logical devices as well as the *functional elements* can be hierarchically structured.

In some cases the hierarchy of device, resource (module/ logical device) and *functional element* can be collapsed, e.g. if a device has only one resource with a single *functional element* it may only provide a parameter list.

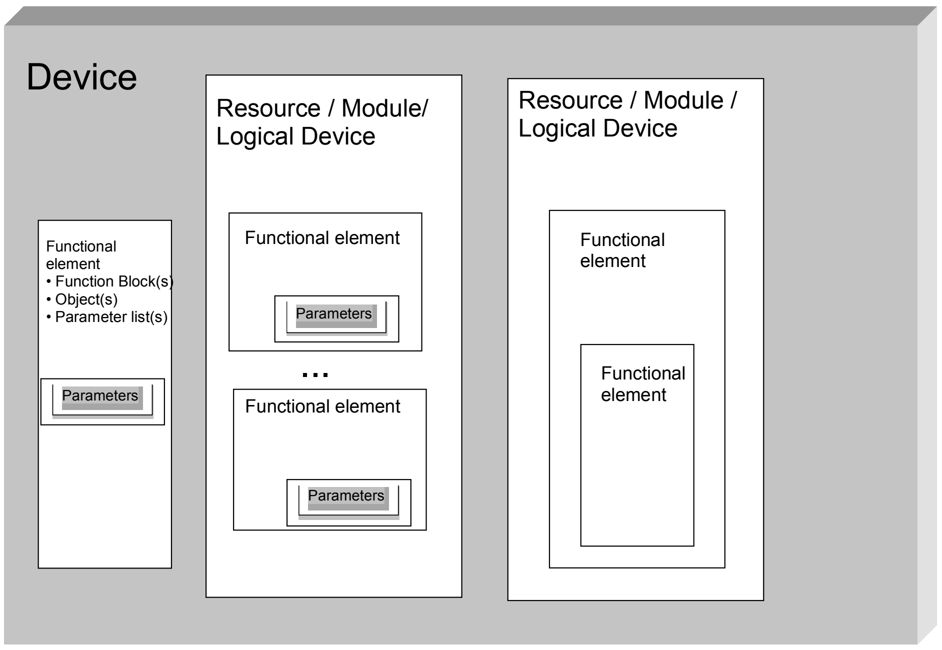


Figure 4 – Example of a modular view of a device's hardware and software structure

The device structure shown above is formally defined in the UML class diagram in Figure 5 (see Annex F).

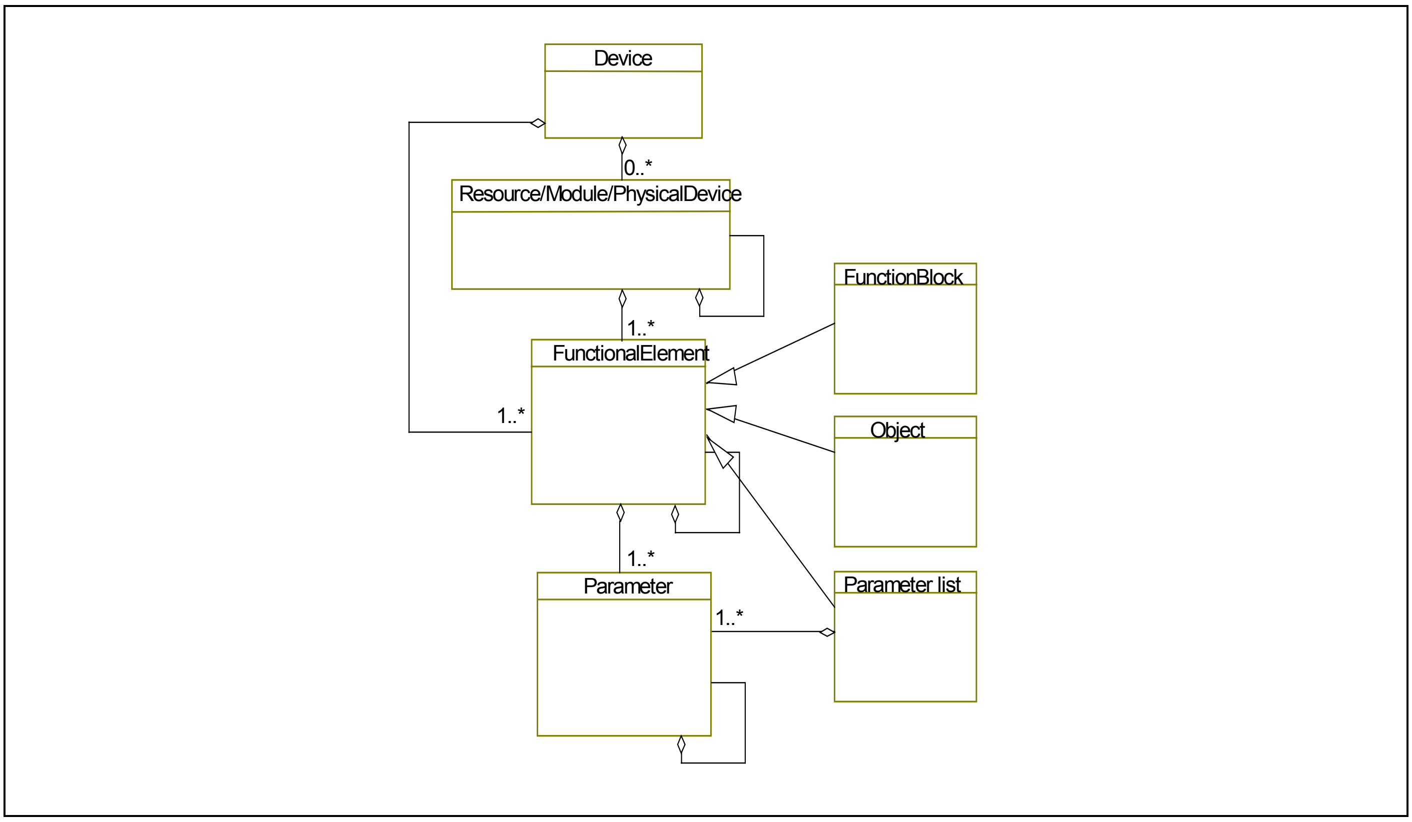


Figure 5 – Example of a device structure class diagram

5.4 Interface model

A device may also be modelled from an interface point of view, if its internal structure is not relevant. The structure of the interface can be derived from the roles the device plays during its operation. Figure 6 shows the following interfaces:

Process interface, which represent the attachment to the process,

Diagnostics interface, which represent all diagnostics information which is provided by a device,

Parameterisation/configuration interface, which represents all structural and functional adjustments of the device during commissioning, operation parameterisation and maintenance,

Control interface, which represent all data related to the control of other devices or higher control hierarchies.

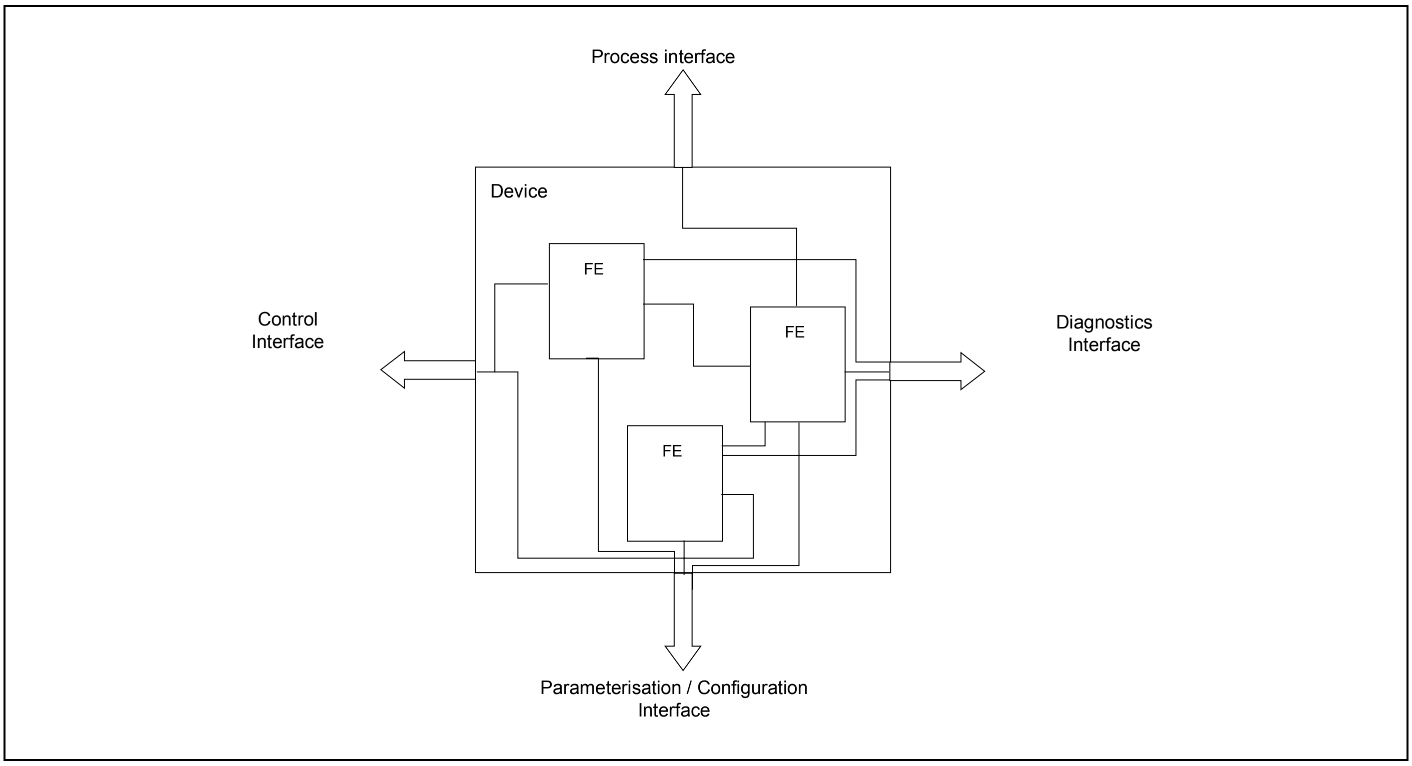


Figure 6 – General interface model of a device

NOTE Control, diagnostics and parameterisation/configuration data is typically accessible via communication interfaces for network connected devices.

6 Profile definition Steps

6.1 Outline

The developing of field device profiles is a 6 steps process as illustrated in Figure 7. During this process all relevant information has to be collected in the filled-in (completed) profile template. These steps are described in detail in the clauses 6.2 to 6.6. A profile document may provide additional explanations and background information.

The first step of each device profile development is the definition of the profile scope and its device classification. This means that the topic of interest has to be clarified by defining which device classes are the subjects of the profile. Additionally, it is very helpful to show the roles of the chosen device classes in the automation system. This helps to take decisions during the specification work.

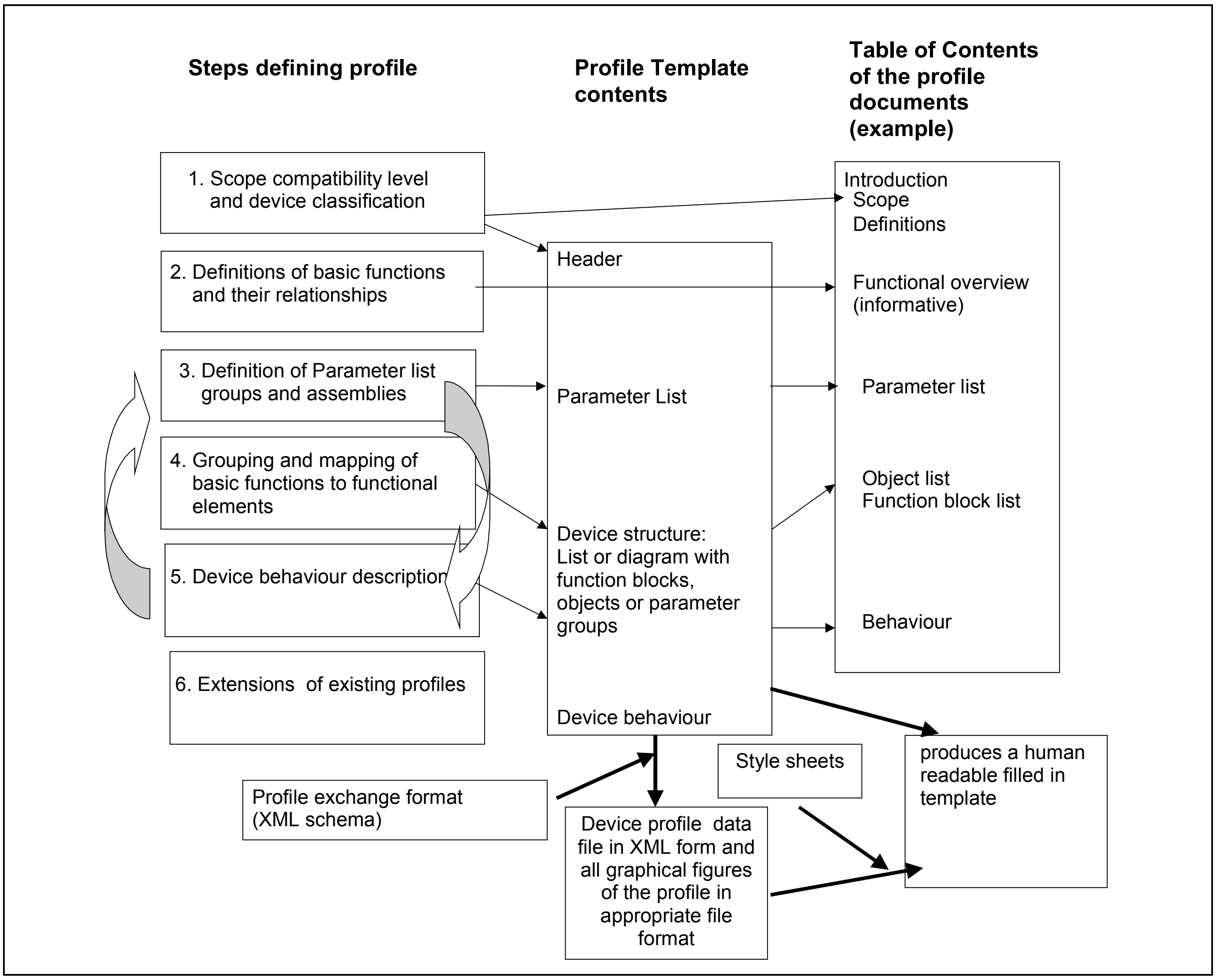

Steps defining profile
Profile Template contents
Table of Contents of the profile documents (example)
1. Scope compatibility level and device classification
2. Definitions of basic functions and their relationships
3. Definition of Parameter list groups and assemblies
4. Grouping and mapping of basic functions to functional elements
5. Device behaviour description
6. Extensions of existing profiles
Header
Parameter List
Device structure: List or diagram with function blocks, objects or parameter groups
Device behaviour
Introduction
Scope
Definitions
Functional overview (informative)
Parameter list
Object list
Function block list
Behaviour
Profile exchange format (XML schema)
Device profile data file in XML form and all graphical figures of the profile in appropriate file format
Style sheets
produces a human readable filled in template

Figure 7 – Profile definition steps

NOTE Other machine readable representations e.g. spread sheets are also appropriate

The second step determines the basic functions, i.e. the functionality of the device classes that are in the scope of the profile. An according functional overview is dedicated to the users of the profile to understand the main functionality which is accessible over the communication network but also to the profile writer to refer to the basics during the profile development process.

The third step defines the parameter list which contains all parameters of the device that are accessible via the communication system. The parameter list is recorded in the parameter list section of the profile template. The parameter list defines the parameters application specific characteristics, excluding communication specific aspects. Profile groups may decide to stop the profile definition work at this level.

The fourth optional step groups parameters and related functions into so called *functional elements*. The resulting *functional elements* are characterised by parameters and behaviour. This step transfers the functional overview developed in the second step into the visible device structure consisting of function blocks or objects which is recorded in the device functional structure section of the profile template. Details are described in clause 8.

The fifth optional step describes the behaviour of the device and/or *functional elements*. This is done separately and recorded in the device behaviour section of the profile template.

It will be helpful to repeat iteratively the execution of the steps 3 to 5 to assure a complete analysis of the device.

The sixth optional step may be applied for the extension of an existing device profile to develop specific members of a device family. This may be done also directly during the execution of the steps 1 to 5.

NOTE Extensions may be made either by adding parameters and *functional elements* to the base profile, or by requiring support of optional items of the base profile.

The profile template is the human readable printed form of the device profile structure and when filled in contains the result of the profile development work. Additional profile documentation may be supplied to provide more detailed information, by extending the information captured in the filled-in template with explanations and additional text and figures.

A machine-readable representation of the device profile is also necessary for use by engineering tools. XML provides among others (e.g. spread sheets) an appropriate technology (Figure 7). The use of XML is recommended.

If XML is used the profile template structure should be represented as an XML schema. A device profile, i.e. the filled in profile template is then represented by an XML file. Style sheets can be used to generate out of the device profile XML files different formats which then can be used by browsers, text processors or other tools. The details on how to use this technology is up to the profile writers or their organisations.

6.2 First step: Scope, compatibility levels and device classification

6.2.1 Overview

The information flow within a system between devices goes from the information detection of the process (transmitters, input devices) through the information processing (control) to the information use at the process (actuators, drives, output devices). A typical automation configuration is given in Figure 8. The information flow is carried out by a signal flow along a chain of functions. Human Machine Interfaces (HMI) and information processing for maintenance and technical management accompany this chain. The device classification step chooses those devices that shall be under profile development. Additionally device identity information (such as device family information and manufacturer) are collected and defined in the first step.

6.2.2 Compatibility levels

6.2.2.1 Background

The following considerations should be done at the beginning of a profile development process. There are certain device profile groups which are dealing with different device classes such as proximity switches, transmitters or drives. Device products take use of the device profiles and add certain manufacturer specific functionality. The device manufacturer connects his device with several communication systems which are also based on communication profiles provided by communication system organisations ((AX, AY, AZ), (BX, BY, BZ) or (CX, CY, CZ)). From the system point of view there are products based on different device profiles using one certain communication system ((AX, BX, CX), (AY, BY, CY), or (AZ, BZ, CZ)). Connecting devices to a system there are N:M combinations of device profiles and communication systems which are increased by manufacturer additions also.

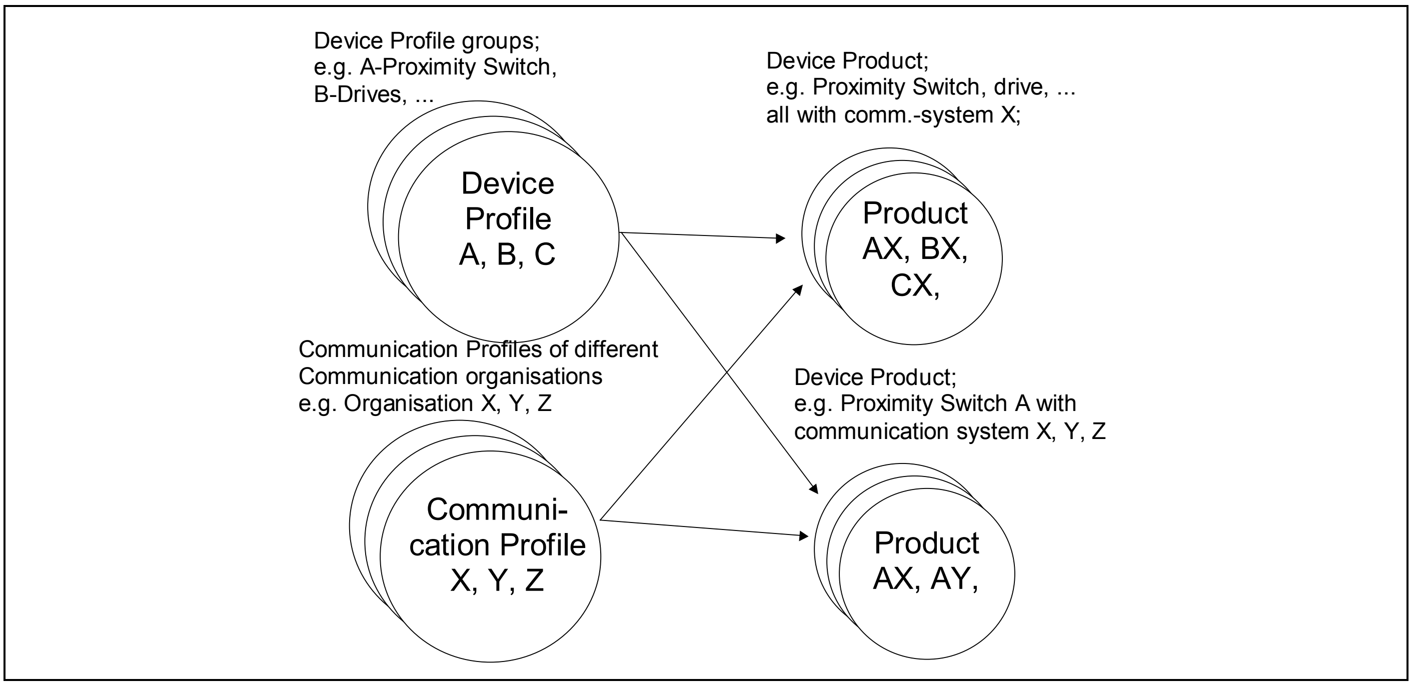
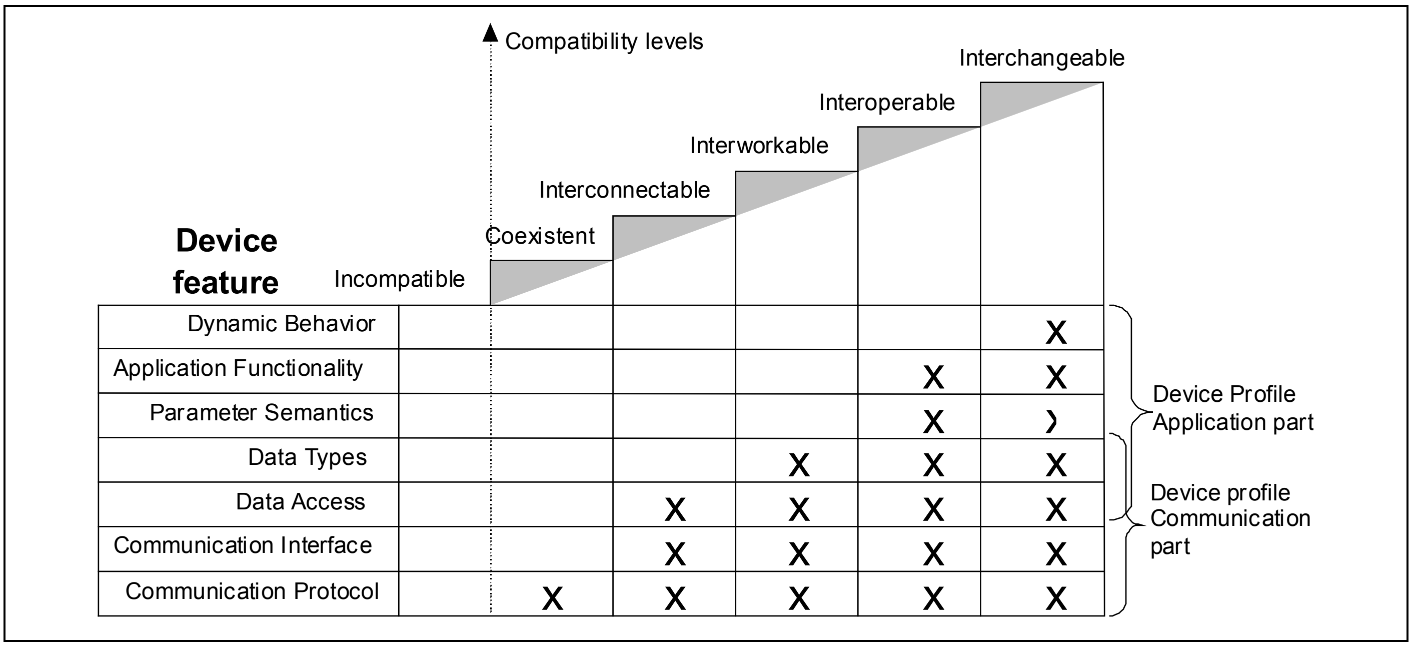

Device Profile groups;
e.g. A-Proximity Switch,
B-Drives, ...
Device Profile A, B, C
Device Product;
e.g. Proximity Switch, drive, ...
all with comm.-system X;
Product AX, BX, CX,
Communication Profiles of different
Communication organisations
e.g. Organisation X, Y, Z
Communi-cation Profile X, Y, Z
Device Product;
e.g. Proximity Switch A with
communication system X, Y, Z
Product AX, AY,

Figure 8 – Relations between profiles and products

There are certain degrees of compatibility and according degrees of co-operation between profiles based devices (compatibility levels). Compatibility levels are applicable for various roles of a device e.g. control, diagnosis, parameterisation/configuration and even applicable to subsets of its functionality. That means that one device can have different compatibility levels regarding different interfaces to the system. The levels are dependent on well-defined communication and application device features.


Compatibility levels
Device Profile Application part
Device profile Communication part

Device feature	Incompatible	Coexistent	Interconnectable	Interworkable	Interoperable	Interchangeable
Dynamic Behavior						X
Application Functionality					X	X
Parameter Semantics					X	)
Data Types				X	X	X
Data Access			X	X	X	X
Communication Interface			X	X	X	X
Communication Protocol		X	X	X	X	X

Figure 9 – Levels of functional compatibility

NOTE The communication profile part is not in the scope of this guideline

The device features are defined as follows.

Table 1 – Device application and communication features

Device feature	Description
Device profile communication part	
Communication Protocol (lower layers)	This feature is defined by all protocols of layer 1 to 4 of the OSI reference model, i.e. from the physical medium access to the transport layer protocol.
Communication Interface (upper layers)	This feature is defined by the communication service definition of application layer including the services and the service parameters. Additional mapping mechanisms can be necessary. The dynamic performance of the communication system is part of this feature.
Device profile communication and application part	
Data access	This feature is defined by the “access” parameter characteristic (see Annex B).
Data types	This feature is defined by the “data type” parameter characteristic (see Annex B).
Device profile application part	
Data semantics	This feature is defined by the parameter characteristics: parameter name, parameter descriptions the parameter range, Substitute value of the parameter, default value, persistence of the parameter after power loss and deployment.
Application functionality	This feature is defined by specifying the dependencies and consistency rules within the *functional elements*. This is done in the parameter description characteristics or in the device behaviour section.
Dynamic behaviour	This feature is defined by time constraints that influence the parameter update or the general device behaviour. For example the update rate of a process value can influence block algorithms.

The relation between device features and parameter characteristics are defined in Annex C. Regarding these device application and communication features the following compatibility levels, Incompatibility, Coexistence and Interconnectability at the communication level and Interworkability, Interoperability and Interchangeability at the application level, should be used for the classification of devices:

6.2.2.2 Incompatibility

Two or more devices are incompatible if they can not exist together in the same distributed application.

NOTE Incompatibility can result from differences in application functionality, data semantic, data types, communications interface, or even communications protocols used by the affected devices. Incompatible devices may even interfere with or prevent each other’s proper communication or functioning (possibly even destructively), if placed in the same distributed application network.

6.2.2.3 Coexistence

Two or more devices coexist on the same communications network if they can operate independently of one another in a distributed application or can operate together using some or all of the same communication protocols, without interfering with the application of other devices on the network.

6.2.2.4 Interconnectability

Two or more devices are interconnectable if they are using the same communication protocols, communication interface and data access.

6.2.2.5 Interworkability

Two or more devices are interworkable if they can transfer parameters between them, i.e. in addition to the communication protocol, communication interface and data access, the parameter data types are the same.

6.2.2.6 Interoperability

Two or more devices are interoperable if they can work together to perform a specific role in one or more distributed applications. The parameters and their application related functionality fit together both syntactically and semantically. Interoperability is achieved when the devices support complementary sets of parameters and functions belonging to the same profile.

6.2.2.7 Interchangeability

Unlike the other compatibility levels (which refer to two or more devices working in the same system) interchangeability refers to the replacement of one device with another. Devices are interchangeable for a given role in a distributed application if the new device has the functionality to meet the application requirements.

NOTE Full interchangeability regarding the entire device performance is nearly impossible to achieve. However actual device interchangeability is dependent on the application requirements for this device.

6.2.3 Device classes

There are already various classification overviews for measurement and actuation devices. These activities standardize the structure of device manuals and additionally the semantic of technical terms, e.g. environmental conditions, signal input and others. They point out the relation to already existing international standards and provide at least taxonomy of measurement and actuation devices. The basic example of automation device classification is provided in Annex G. A device class is a set of devices with a defined functional commonality in terms of their *parameters* or *functional elements*.

This step shall result in an agreement on a common scope of the profile, a specific device class or device family, a commonly targeted level of compatibility of the discussed devices and the necessary information for the profile template and documentation. As shown in Figure 7 all relevant information of this step are recorded in the header section of the profile template as shown in clause 7.2.5.

6.3 Second step: Definition of device functions and their relations

The device is described in a top down approach based on a black box model, i.e. starting with the external interface (e.g. process input/output) or connection (e.g. sensor, valve, motor) and the main control input/output (e.g. setpoint, measurement value). This first model is detailed by stepwise refining of the main signal flow between device functions. The degree of details depends on the device class. Different device sub-classes may be introduced at a detailed level. Devices may have certain sub-classes which provide functions for different purposes. These sub-classes should be shown.

The overview of device functions provides the functional structure of the chosen device class (e.g. drives). It is possible that multiple functional diagrams are necessary to cover the device sub-classes (e.g. AC and DC drives) of the profile.

The list of device functions is not part of the profile template.

The third to the fifth step will be carried out iteratively because they are tightly connected.

Functional diagrams (see Figure 10) are the main descriptions of this step that should be accompanied by textual descriptions. Additionally it is recommended to define use cases and scenarios (e.g. using UML as shown in Figure 11) for which the device profiles are defined.

The example in Figure 10 shows a not standardised functional diagram of a power drive system. The purpose is to express the main device functions and their relation which reflect the common view of the device profile group for a specific device class. For example the PDSs (Power Drive System) have hundreds of functions which might be considered. It is not possible to consider all of them. Therefore, a significant choice has to be made.

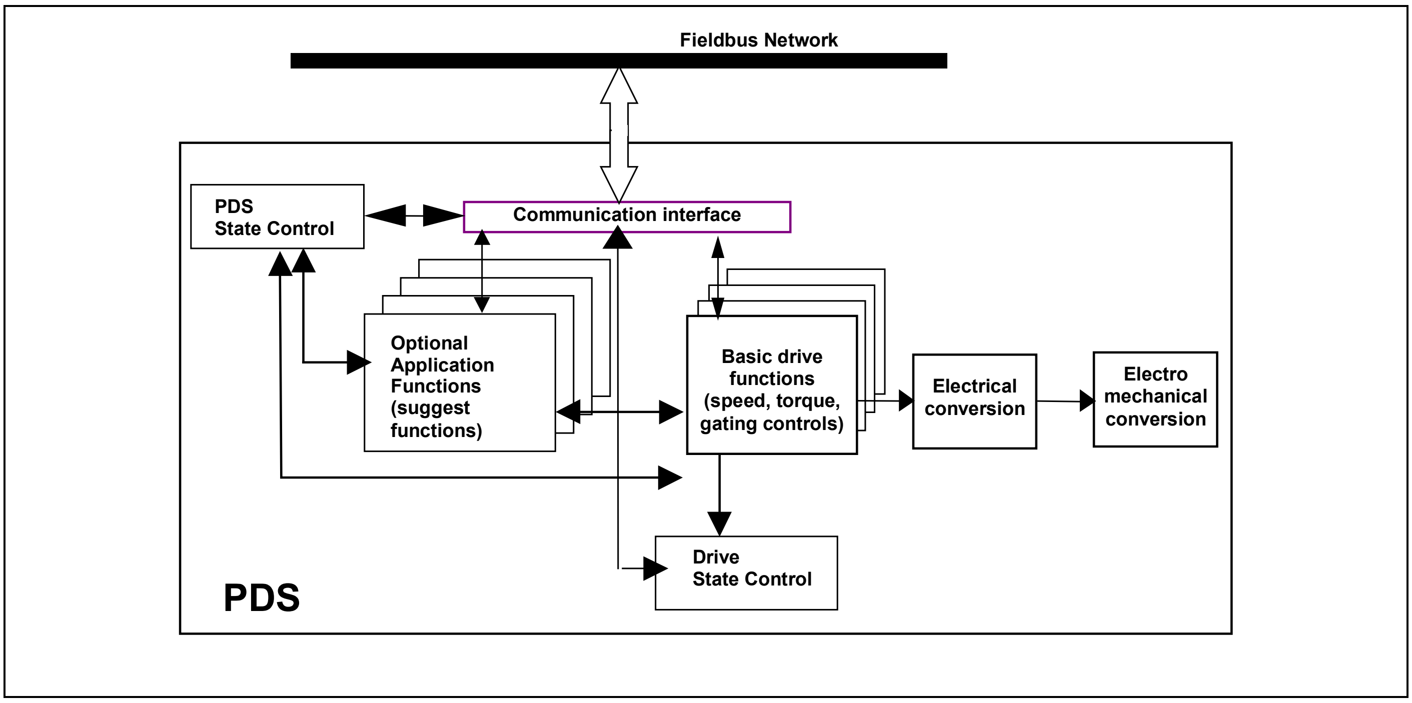


Figure 10 – Example functional diagram of a power drive system (PDS)

6.4 Third step: Parameter list definition

The parameter list defined in this step contains all accessible parameters of a device. The definition of the parameters can follow various procedures, for example:

Derivation of the parameters out of the device functions – see 6.3

Considerations of life cycle aspects - see Annex A

Investigations of the device use cases - see Figure 11

These parameters have characteristics (e.g. name and data type) that allow the user of the device to properly provide, use and display a parameter value. A representative collection of possible parameter characteristics is provided in Annex B. Depending on the functional compatibility level (see 6.2.2) the profile writer is intending to achieve, a certain subset of parameter characteristics has to be defined. The "support" characteristic is a major one which may be used to extend a base profile, e.g. by requiring support of optional parameters of the base profile.

The parameter list is recorded in the parameter list section of the profile template (see clause 7).

Complex devices may have large numbers of parameters. For this purpose, parameter groups may be defined; a parameter group is a logical set of parameters which may be associated with the same function and/or use case of a device. The definition of parameter groups is optional.

A parameter assembly is a set of parameters which is accessible in a single network read or write service. Not all networks support parameter assemblies. Therefore the definition of parameter assemblies is optional.

To define the parameter list as part of the profile template, one possibility is to start with use cases. Use case definition should follow the definition of UML. A use case specifies a sequence of actions that a system can perform, interacting with so called actors (see UML and Figure 11). From the profile definition point of view all phases of the device life cycle have to be considered. Actors can be human beings as well a software and hardware components. Typical actors here are controller, PC based tools and operators which interact with the devices in terms of operation, parameterisation, configuration and maintenance. The analysis of the interactions between the controller and PC tools and the device leads to a list of relevant parameters. Additionally, the interactions need a defined device behaviour which is also part of the device profile.

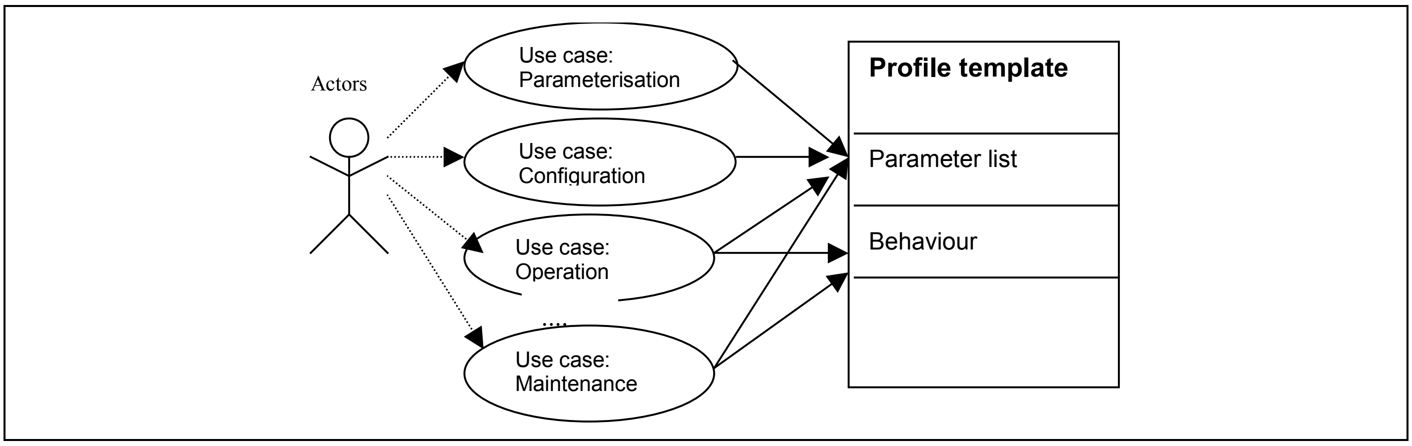


Figure 11 – UML use case examples

Examples for possible role of actors and their actions are listed in Annex A.

Profile groups may decide to stop the profile definition work at this level. If not, then the third to the fifth step may be carried out iteratively because they are tightly connected.

6.5 Fourth step: Grouping of functions to *functional elements*

6.5.1 Description

The functional diagrams of a certain device (see step 2, clause 6.3) give an overview of device functions and their relation to each other. The grouping of the functions to *functional elements* will be done in this profile development step.

In the industrial automation domain there are different approaches on how to model devices. These approaches are described in the annexes:

Parameter List model – see Annex H

Function block model– see Annex I

Object model - see Annex J

The profile writer group has to agree on which model they will use for their profile development. To offer a common view of these three approaches the term *functional element* is introduced in this guideline. A *functional element* is either a parameter or parameter group, a function block, or an object.

The defined functional elements are recorded in the device structure section of the profile template (see clause 7).

The third to the fifth step will be carried out iteratively because they are tightly connected.

6.5.2 Example of a flow transmitter using object model and function block model

Figure 12 and Figure 13 show the structure of a flow transmitter, which is modelled using either objects or function blocks.

NOTE The relationship between the two models is shown in Table 4.

The dotted ellipse of Figure 12 represents the application object class with an internal structure. Each sub-object class is specified in detail in the corresponding profile. The Flow Transducer represents the hardware only and is out of the scope of the corresponding profile. Parameter and Assembly object classes are the interfaces of the application object class with the communication system. Message router and connection object class are communication system specific classes.

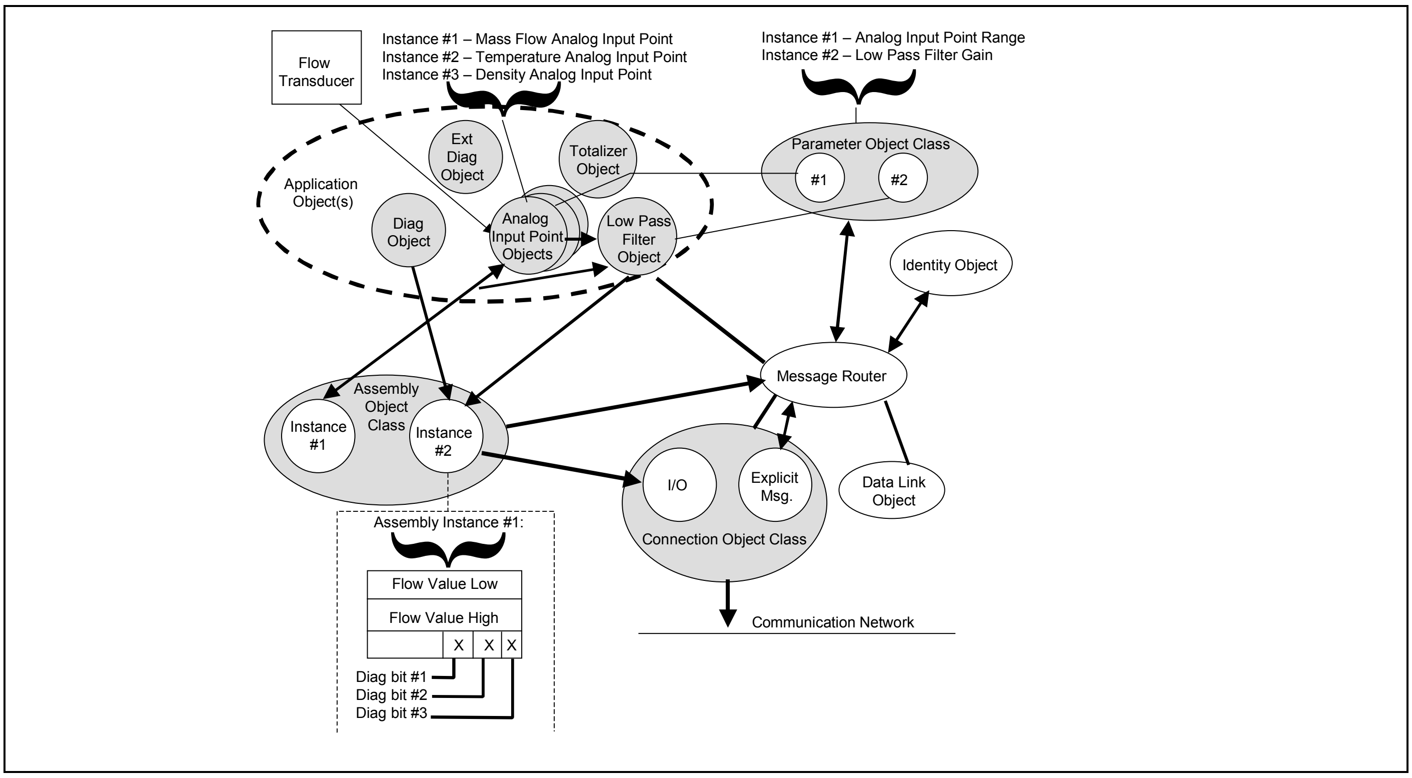


Figure 12 – Device functional structure of a flow transmitter based on the object model (example)

The flow transmitter example based on the function block model shows a detailed structure regarding the internal signal flow (Figure 13). This example models three process values (i.e. mass flow, density and temperature), each with a separate analogue application signal processing. Additionally a totalizer function block counts the mass flow. Communication services may access function block parameters directly or using view function blocks.

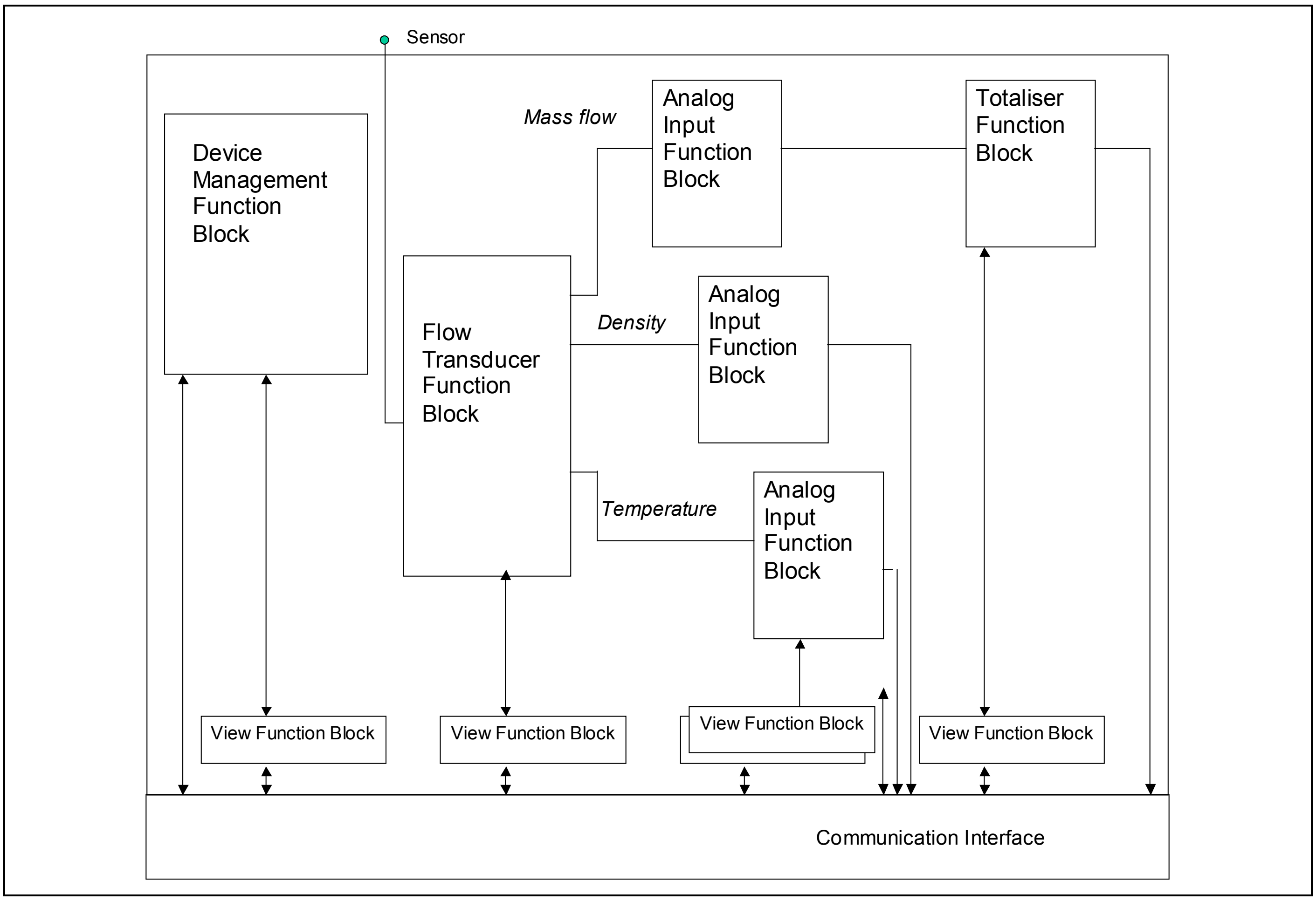


Figure 13 – Device functional structure of a flow transmitter based on the function block model (example)

6.6 Fifth step: Device behaviour description

Dependencies between parameters and resources cannot be easily expressed using parameter characteristics. Each *functional element* may have its own behaviour. Therefore this step addresses the behaviour point of view.

Profile writers choose a relevant subset of the device behaviour. The behaviour of the device and/or function blocks and objects is composed of a set of algorithms and methods respectively. Behaviour is commonly described using the following:

An algorithm in the mathematical sense (e.g. normalize, scaling, filter, invert) which describes how to calculate output data from input data using parameters; one possible description of these algorithms are IEC 61131-3 functions

A sequence of algorithms including process and communication interactions (e.g. alarm handling, calibration, start up phase, time dependent start of system self-test or sensor cleaning process)

A state machine (example states are the operation modes of a device like running, ready, stop as shown in Figure 14 and Table 2)

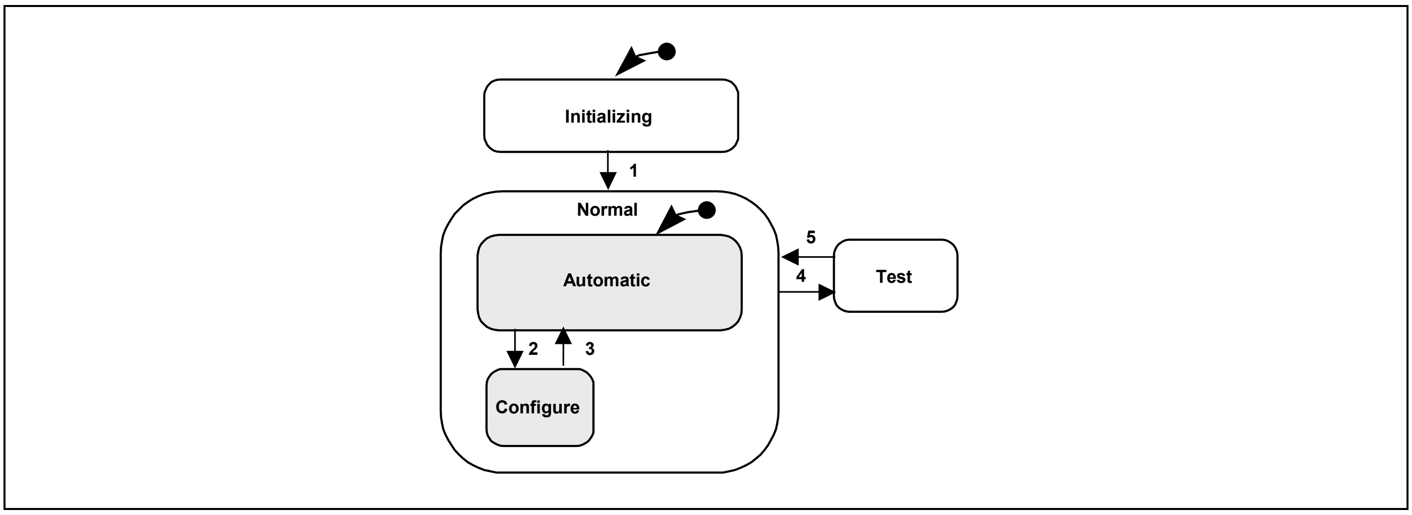


Figure 14 – Example of device behaviour as statechart diagram

Table 2 – Example of a device behaviour as state transition table

STATE NAME	STATE DESCRIPTION
Initialising	Initial state of the device upon power-up. The device is not yet available for normal operation
Normal	The device is available for automatic operation
Automatic	“Presence” and “Alarm” parameter values are available to the network
Configure	When in this state, the device can be commanded by the “Operate mode” parameter to alter its operation accordingly (light or dark signal indicates presence) The device will not perform its normal sensing operations in this state – the “Presence” and “Alarm” parameter values should not be read by the network
Test	The device does not perform its normal sensing operations. The “Presence” and “Alarm” parameter values are set to one (1)

TRAN-SITION	SOURCE STATE	TARGET STATE	EVENT
01	Initialising	Normal	Device initialised and ready for normal operation
02	Automatic	Configure	“Device mode” parameter is commanded to change from zero (0) to one (1), or service “Set configure mode” is invoked
03	Configure	Automatic	“Device mode” parameter is commanded to change from one (1) to zero (0)), or service “Set automatic mode” is invoked
04	Normal	Test	“Test” parameter is commanded to change from zero (0) to one (1)), or service “Enter test mode” is invoked
05	Test	Normal	“Test” parameter is commanded to change from one (1) to zero (0)), or service “Exit test mode” is invoked

The result of this step is the detailed behaviour description of the device and/or *functional elements* that are recorded in the device behaviour section of the profile template.

NOTE If a device is user programmable, its behaviour cannot be completely described in the profile. However profile writers may agree on general common functions like Start, Stop and Reset.

The third to the fifth step will be carried out iteratively because they are tightly connected.

6.7 Sixth step (optional): extensions of existing profiles

This optional step is necessary if a defined set of profiles or devices is derived out of a root or manufacturer device profile. The derived profile may extend these profiles by:

Adding parameters, behaviour, *functional elements* items

Requiring support of optional items of root or manufacturer device profiles

Steps 1 to 5 may have to be reconsidered. This process leads to a set of related device profiles.

7 Profile templates

7.1 General

As explained in the introduction of this guideline and in Figure 1, profile writers provide a common representation of the network industrial devices. This guideline recommends a profile template for documenting that representation. This profile template serves as a form that, when filled-in by profile writers, becomes a human readable device profile. A filled-in profile template is the result of the profile development procedure described in clause 6. The filled-in template may be exchanged using a profile exchange language such as XML (see 6.1).

7.2 Profile Template structure

7.2.1 Overview

The profile template is organized in sections. The following sections are recommended:

The header section contains the results of the first profile development step “scope, device classification”, together with revision information.

The parameter list section contains the results of the third profile development step. There are 3 sub-sections: parameters with their characteristics, parameter groups and parameter assemblies.

The device functional structure section contains the result of the second, fourth and fifth profile development steps. There are 2 sub-sections: the functional structure based on the functional elements (defined in the fourth step) and the device behaviour (defined in the fifth step).

These profile template sections are detailed in 7.2.2 to 7.2.4. It is the responsibility of the profile writer group to define the details of the profile template sections. An example of such a profile template is provided in 7.2.5, showing all the profile sections above.

7.2.2 Header

The contents of the header are the result of the first profile development step.

The header section provides device profile identification. The header makes it clear to the reader of the profile that he has the correct device profile.

If a profile is defined for a class of devices (e.g. a profile defined by an IEC product committee, called “root profile” in Figure 1), the contents of the header provides an unambiguous identification of this profile, including the identifier assigned to this profile by the profile writer organization (Device profile ID), profile revision (Device profile version) and profile release date (Device profile release date). Fields for additional device information may be provided.

If a profile is defined for a specific device (e.g. a profile defined by a manufacturer, called “manufacturer’s profile” in Figure 1), the header provides additionally an unambiguous identification of this specific device. This identification information includes for example device’s name, catalogue number, manufacturer, and version. Fields for additional device information may be provided.

7.2.3 Parameter list

7.2.3.1 Parameters

The contents of the parameter list are the result of the third profile development step.

The “Parameters” section of the template provides a list of all the device’s parameters that are accessible in the device through the network, together with their relevant characteristics.

A separate named field should be provided for each specified characteristic.

An example of parameter characteristics is included in the example template of sub-clause 7.2.5.

7.2.3.2 Parameter groups

The parameter group definitions are the result of the third profile development step.

7.2.3.3 Parameter assemblies

The parameter group definitions are the result of the third profile development step.

7.2.4 Device functional structure

7.2.4.1 Functional elements

The device functional structure definitions are the result of the second and fourth profile development step.

The template contains the description of the functional structure using a *functional element* list and an optional functional structure diagram showing the relationships between the *functional elements*.

Simple devices may only consist of a single *functional element*.

Complex devices may consist of a collection of *functional elements*.

7.2.4.2 Device Behaviour

7.2.4.2.1 State machines

The device functional structure definitions are the result of the second and fifth profile development step.

It is recommended to describe the behaviours of the device or *functional element*. Good practice is to do this in terms of a state chart; an example is shown in Figure 14. In this case the profile template in 7.2.5 contains a state machine section for specifying a state chart diagram and a state transition table. It is recommended to specify both diagram and table because the diagram is suitable for a quick human overview and the table is suitable for a mapping to a machine readable format. The descriptions of state machines and state chart diagrams should follow the UML specification.

7.2.4.2.2 Mathematical and procedural behaviour

The device behaviour definitions are the result of the fifth profile development step.

Behaviour of *functional elements* may be expressed using mathematical equations, procedural and consistency rules among parameters and conditions. IEC 61131 should be preferably used to describe this behaviour.

7.2.5 Template form

Table 3 shows an example of a representation format for the profile template.

NOTE This example is derived from the template specified in IEC/TS 61915:2003.

Table 3 – Filled in template of a device profile (example)

DEVICE PROFILE HEADER

Device profile ID	Device profile version	Device profile release date	Device profile description
Chd	V.53	2003-02-27	This is an example profile.

PARAMETER LIST [optional]

NOTE 1 A parameter may be assigned to multiple parameter groups
NOTE 2 Annex B provides a list of possible parameter characteristics

PARAMETERS

Parameter name	Data type	Access	Range	Support	Parameter description
Chv	Boolean	r/w	0,1	Mandatory	This is an example parameter

PARAMETER GROUPS

Group name	Number of elements	Group description
JPWG	2	This is an example group

Member names:
HPO HW

PARAMETER ASSEMBLIES

NOTE Parameter assemblies are defined for communication purposes only, e.g. for periodic data exchange. There are independent of the parameter groups of the parameter list.

Parameter assembly name	Access	Support	
BD	r/w	Mandatory	

Byte and Bit structure

DEVICE FUNCTIONAL STRUCTURE [optional]

FUNCTIONAL ELEMENTS

FUNCTIONAL STRUCTURE DIAGRAM
Provide a function block diagram or object diagram, examples are shown in Figure 12 and Figure 13.

FUNCTIONAL ELEMENT LIST

Functional element name	Support	Description
IL	Mandatory	This is an example functional element.

DEVICE BEHAVIOUR [optional]
NOTE A behaviour description may be provided for the entire device and/or for functional elements

STATECHART DIAGRAM
Provide state chart diagram here (example is shown in Figure 14)

STATE TRANSITION TABLE (example is shown in Table 2)

State name	State description

Transition	Source state	Target state	Event

MATHEMATICAL OR PROCEDURAL BEHAVIOUR
(see sub-clause 6.6)

8 Device models

8.1 Mapping of ISO Device Profile Classes

This guideline is based on the device profile definition of the ISO 15745-1. The corresponding device profile class diagram is shown in Figure 15.

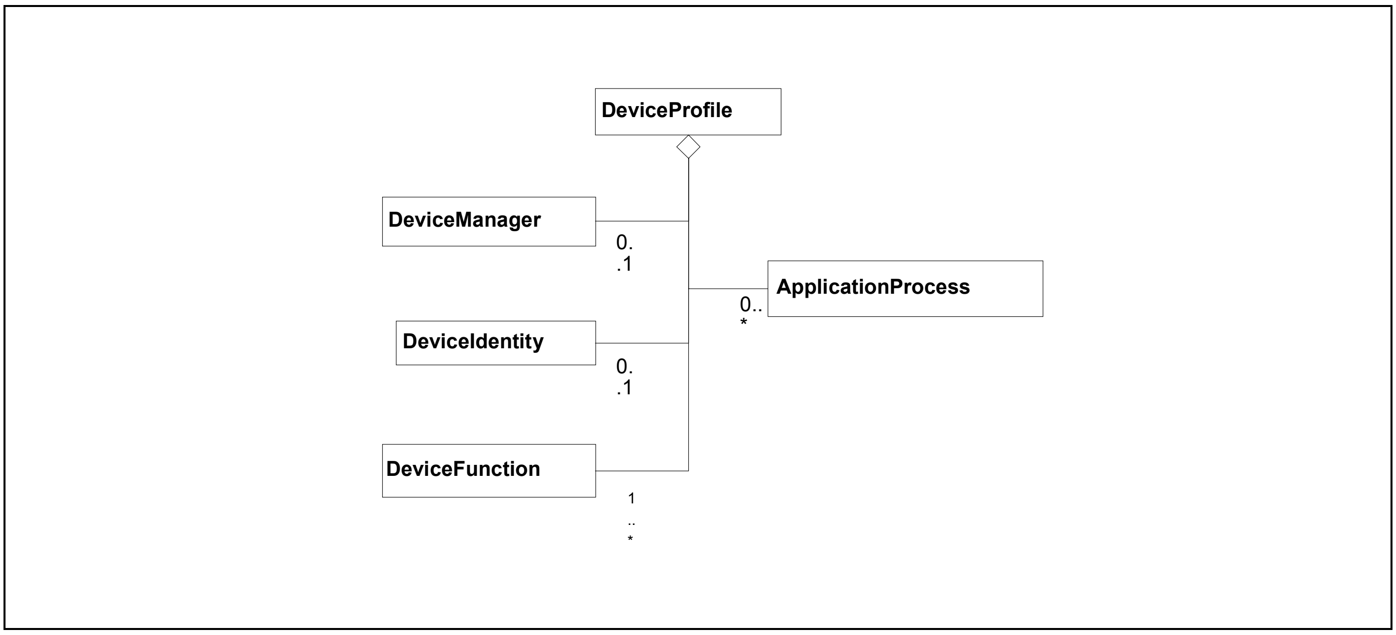


Figure 15 – ISO 15745-1 Device Profile Class Diagram

The following class definitions are given in ISO 15745-1:2003.

Device identity
The device identity object contains attributes which uniquely identify the device. Examples of such attributes are the manufacturer's identification, part number, revision, location of storage of additional information, and indication of the number and type of additional objects within the device.

Device manager
The device manager object represents the set of attributes (e.g. revision of the device identity object) and services (e.g. reset, configure/run mode, retrieval of device manager object attributes) used to configure and to monitor a device integrated into the application system.

Device function
The device function object describes the intrinsic function of a device in terms of its technology (e.g. mechanical limit switch, proximity sensor, ultrasonic sensor). The device function object differentiates the technology of the device from the application of the device. Examples of device function objects are analogue current input in milliamps, and discrete voltage output in volts.

Application process
The application process object represents a set of attributes and services that correspond to the application requirements captured in the attributes and services of the associated process profile. The application process object therefore describes the behaviour of the device in terms of the application, independent of the device technology.

EXAMPLE An example of an application process object is a section of code within a device that detects, validates, and reports the presence (or absence) of a part, independent of the device technology being used. An infrared photoelectric sensor, a capacitive proximity switch, or a piezoelectric pressure device can meet the application requirement represented by the same application process object.

A simple device may contain one application process object. A complex device may contain one or more application process objects. In a distributed system, one application process object may span a number of devices.

The ISO 15745 device profile classes are seen from an abstract point of view. In principle there are multiple mappings to it as shown in Figure 16.

The parameter list model (b) in Figure 16 represents the simplest mapping of the device model. It is described in Annex H. Simple devices may be fully described using parameter lists only. Parameters may be organised into parameter groups using the ISO 15745-1 device profile classes.

The parameter list model may be extended using either the function block model (c) or the object model (d) shown in Figure 16. The function block model is described in Annex I and the object model in Annex J.

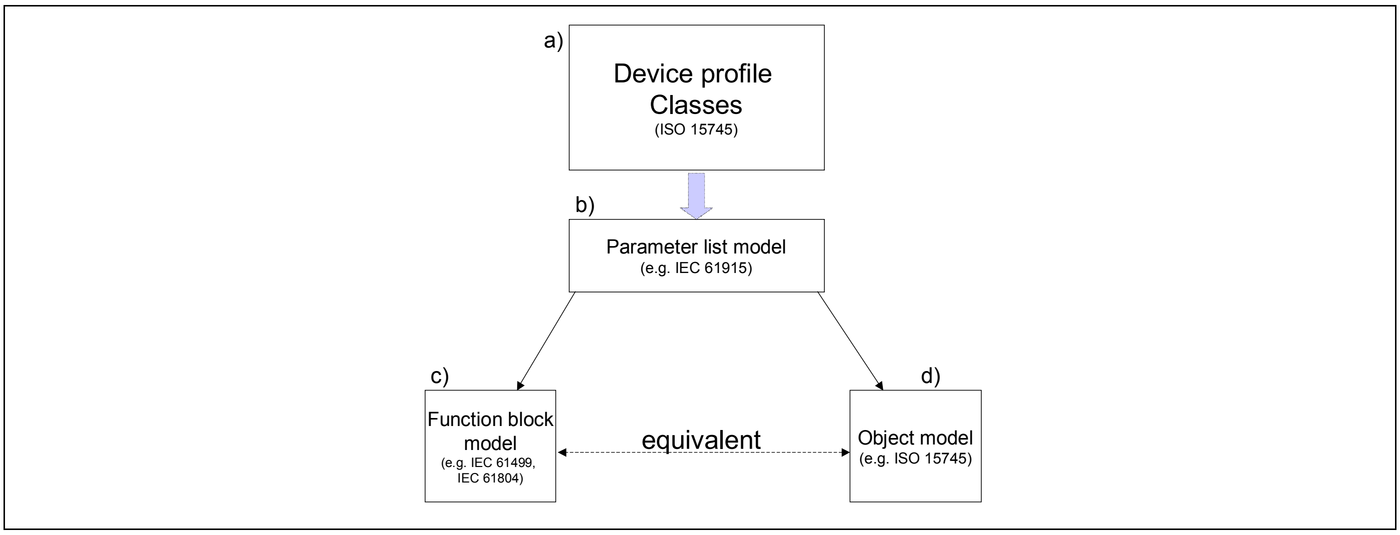


Figure 16 – Device profile models

8.2 Comparison of function block and object device model

The function block model and the object model are equivalent and follow the object oriented concepts in terms of:

Encapsulation of data and functions

Instantiations of classes to objects; these are also called types and instances

Class/object hierarchies

Equivalence between the block and object device model is shown in Table 4.

Table 4 – Equivalence between function block and object device models

Device Profile objects ISO 15745-1	Parameter list model elements	Function block model elements	Object model elements
Application process object	Parameter group	Application function block View	Application specific (s) object Parameter object Assembly object
Device function object		Transducer function block	Application specific (s) object
Device identity		Device management function block	Device identity & management
Device manager			
-	Parameter	Function block parameter	Attribute of a parameter or application specific object

Annex A (informative) Roles of the device in the life cycle

All phases of the device life cycle should be considered during the device profile development process, more specifically when defining the parameter list (see 6.4). Examples of such phases are listed below.

Commissioning, maintenance and servicing phases include the following tasks:

Configuration
Determines the hardware (e.g. installed module) and software (e.g. instantiated objects/blocks) structure of the device

Parameterisation
Determines the initial / default and process related values of the device parameters.

Working point adjustments / calibration
Determines the relation between the real process (i.e. physical, biological, chemical) values and the measurement and actuation variables.

Structural information for design and engineering
Provides information on the functional elements (e.g. parameters, function blocks, objects) implemented in the device. This information may be accessed via a communication system by tools or other devices.

Commands and Management functions
Allow operators or the device itself to change device states and operation modes.

Firmware update

Device information
Allows access to device identification and configuration information (e.g. catalogue number, manufacturer’s URL for additional information)

Diagnostics
Provides device status and faults

Statistics/trends
Provides device activity history

Operation phases include the following tasks:

Control application
Provides measurement values, gets actuation set points and exchange other information with other devices or a control program.

Visualization / HMI
Provides measurement values, gets actuation set points and exchange other information with operator for visualization and interactions.

Annex B (informative) Collection of parameter characteristics

Table B.1 provides a representative collection of parameter characteristics which may be used by the profile writer to specify the details of the parameter fields in the parameter list section of the profile template (see 6.4 and 7.2.3.1).

Table B.1 – Collection of parameter characteristics

Parameter characteristic	Conceptual Data type	Description
ID	Implementation specific	Identifier of a particular parameter. The ID has to be unique in a device (see NOTE 1).
Parameter Name	String	Name of a particular parameter (see NOTE 1).
Description	String	Textual description of parameter purpose (see NOTE 2).
Data type	Enumeration	IEC 61131-3 data types are preferred.
Grouping level	Enumeration	*Simple* - basic data type *Array* - collection of data items of the same type *Structure* - collection of data items of different data types
Number of Bytes	Numeric	Number of bytes of the parameter values.
Access timing	Enumeration	Specification of the dynamic characteristics like Periodic changes (cyclic) Episodic changes (acyclic) Change driven (event driven) The way to interact within a system is communication system dependent.
Access direction	Enumeration	Specification whether a parameter can be read and/or written: Read only Read/Write Write only (e.g. password parameter) The way to interact within a system is communication system dependent.
Persistence	Enumeration	Specification whether or not the value is retained in device memory after the device's power was lost. volatile non-volatile
Value range	Numeric	Range of supported values and/or list of supported values (see NOTE 2).
Coding	String	Enumeration of supported parameters values and their usage (see NOTE 2). EXAMPLE 0: OFF; 1: ON.
Default value	Corresponds to data type parameter characteristic	Defines the value that shall be contained in the parameter upon device delivery. If the profile does not specify a default value a manufacturer specific value may be implemented.
Substitute value	Corresponds to data type parameter characteristic	Defines a specific value of the parameter that is provided to the application program in certain device operating states (e.g. device fault).
Engineering unit	String or enumeration	Defines the engineering unit (if relevant) of the parameter. A list of engineering units is provided in Annex E
Offset	Numeric	The offset element specifies an offset which is added to an actual value to form a scaled value. Engineering value= (parameter value + offset) * multiplier

Multiplier	Numeric	Scaling factor by which an actual value is multiplied to form a scaled value Engineering value= (parameter value + offset) * multiplier
Constraint	String	Constraints between parameters, for modelling certain dynamic device behaviour (e.g. disable, enable, change another parameter depending on certain values of this parameter).
Support	Enumeration	Defines whether or not the parameter has to be implemented in the device. Optional: parameter implementation is possible but not required. Mandatory: parameter implementation is required. Conditional: parameter implementation is required if one or more other optional parameter(s) is (are) implemented. These parameters are specified using "Conditional Support".
Conditional Support	String	If Support = Conditional, the optional parameter(s) is (are) specified via this characteristic.
NOTE 1 The usage of parameter ID and parameter name for parameter keying depends on the selected device model. NOTE 2 Value range and coding contents may be combined within a single field or described in the description characteristic field. NOTE 3 The data types used in this table are conceptual types, i.e. they describe the kind of physical signal, not the implementable data type. Mapping onto implementable data types (e.g. IEC 61131-3) is required in actual profiles.		

Annex C (informative) Compatibility level details

Table C.1 shows the relations between device features and parameter characteristics (Table B.1). The rows of the table correspond to the parameter characteristics which are listed in Annex B. The columns correspond to the device features (see 6.2.2).

Table C.1 – Relation between parameter characteristics and device features

Parameter characteristics	**Device features**						
	Comm. protocol	**Comm. interface**	**Data access**	**Data types**	**Parameter semantics**	**Applicatio n functional ity**	**Dynamic behaviour**
ID			X				
Parameter name					X		
Description					X		
Data type				X			
Grouping level				X			
Number of bytes				X			
Access timing			X				X
Access direction			X				
Persistence						X	
Value range					X		
Coding					X	X	
Default value					X		
Substitute value						X	
Engineering unit						X	
Offset						X	
Multiplier						X	
Constraint						X	
Support					X	X	
Conditional support						X	

Annex D (informative) Data Type

The data types of the device parameters need to be defined in the device profile. Table D.1 shows equivalence between data types used by various standards.

A proposed subset of data types is suggested which is marked as grey fields in the table.

Table D.1 – Data types

Bits	**IEC 61131-3:2003** subclause 2.3	**IEC 61158-5:2003** clause 5	**OPC**	**"C"**
1	BOOL	Boolean	Boolean	-
8	BYTE	BitString8	Byte	Unsigned char
16	WORD	BitString16	-	Unsigned short
32	DWORD	BitString32	-	Unsigned long
64	LWORD	BitString64	-	-
8	SINT	Integer8	Byte	Char
16	INT	Integer16	Short	Short
32	DINT	Integer32	Int	Long
64	LINT	Integer64	Long	-
8	USINT	Unsigned8	unsignedByte	Unsigned char
16	UINT	Unsigned16	unsignedShort	Unsigned short
32	UDINT	Unsigned32	unsignedInt	Unsigned long
64	ULINT	Unsigned64	unsignedLong	-
32	REAL	Float32	Float	Float
64	LREAL	Float64	Douple	Double
8 x n ASCII	-	VisibleString (ISO/IEC 646)	String	Array of char
8 x n	STRING	OctetString (ByteString)	-	Array of char
-	TIME (duration)	See NOTE 2	Duration	-
-	DATE	See NOTE 2	Date	-
-	TIME_OF_DAY	See NOTE 2	-	-
-	DATE_AND_TIME	See NOTE 2	DateTime	-
-	ARRAY	See NOTE 2	ArrayOfxx	[ ]
-	STRUCT	See NOTE 2	-	Struct
-	Derived data types	-	-	typedef

NOTE 1 Only a subset of defined data types are listed here.

NOTE 2 Implementation of this data type is technology dependent.

NOTE 3 The shaded rows are recommended to be used.

Annex E (informative) Engineering Unit

Table E.1 lists example values for the engineering unit parameter characteristic in Annex B.

NOTE Engineering units are also defined at http://www.physics.nist.gov/Pubs/SP811/cover.html.

Table E.1 – Engineering Units - Examples

Physical dimension	Unit	Abbreviation	Exchange format (see NOTE)
Length, displacement	Meter	m	m
	Millimeter	mm	mm
	Kilometer	km	km
	Micrometer	µm	um
Area	Squaremeter	m^2	m**2
	Squaremillimeter	mm^2	mm**2
	Squarekilometer	km^2	km**2
Volume	Cubicmeter	m^3	m**3
	Litre	l	l
Time	Second	s	s
	Minute	min	min
	Hour	h	h
	Day	d	d
	Millisecond	ms	ms
	Microsecond	µs	us
Force	Newton	N	N
	Kilonewton	kN	kN
	Meganewton	MN	MN
Pressure	Pascal	Pa	Pa
	Kilopascal	kPa	kPa
	Millibar	mbar	mbar
	Bar	bar	bar
Mass	Kilogramme	kg	kg
	Gramme	g	g
	Milligramme	mg	mg
	Ton	t	t
Energy	Joule	J	J
	Kilojoule	kJ	kJ
	Megajoule	MJ	MJ
	Watt hour	Wh	Wh
	Kilowatt hour	kWh	kWh
	Megawatt hour	MWh	MWh
Apparent power	Voltampere	VA	VA
	Kilovoltampere	kVA	kVA
	Megavoltampere	MVA	MVA
	Millivoltampere	mVA	mVA
Rotation speed	1/second	s^{-1}	s**-1
	1/minute	min^{-1}	min**-1
	1/hour	h^{-1}	h**-1
Angle	Radian	rad	rad
	Second	“	“
	Minute	‘	‘
Velocity	Meter/Second	m/s	m/s
	Millimeter/Second	mm/s	mm/s
	Millimeter/Minute	mm/min	mm/min
	Meter/Minute	m/min	m/min
	Kilometer/hour	km/min	km/min
	Millimeter/hour	mm/h	mm/h
	Meter/hour	m/h	m/h
	Kilometer/hour	km/h	km/h
Volume flow	Cubicmeter/Second	m^3/s	m**3/s
	Cubicmeter/Minute	m^3/min	m**3/min
	Cubicmeter/hour	m^3/h	m**3/h
	Liter/Second	l/s	l/s
	Liter/Minute	l/min	l/min
	Liter/hour	l/h	l/h

Physical dimension	**Unit**	**Abbreviation**	**Exchange format** (see NOTE)
Mass flow	Kilogramme/Second Gramme/ Second Tonne/ Second Gramm/Minute Kilogramme/Minute Ton/Minute Gramme/hour Kilogramme/hour Ton/hour	kg/s g/s t/s g/min kg/min t/min g/h kg/h t/h	kg/s g/s t/s g/min kg/min t/min g/h kg/h t/h
Torque	Newtonmeter Kilonewton meter Meganewton meter	Nm kNm MNm	Nm kNm MNm
Temperature	Kelvin Degree Celsius Degree Fahrenheit	K °C °F	K C F
Temperature difference	Kelvin	K	K
Entropy	Joule/(Kelvin*kg) kJ/(K*kg) MJ/(K*kg)	J/(K*kg) kJ/(K*kg) MJ/(K*kg)	J/(K*kg) kJ/(K*kg) MJ/(K*kg)
Enthalpy	Joule/Kilogramme Kilojoule/Kilogramme Megajoule/Kilogramme	J/kg kJ/kg MJ/kg	J/kg kJ/kg MJ/kg
Electrical voltage	Volt Kilovolt Millivolt Mikrovolt	V kV mV µV	V kV mV uV
Electrical current	Ampere Milliampere Kiloampere Microampere	A mA kA µA	A mA kA uA
Electrical resistance	Ohm Milliohm Kiloohm Megaohm	Ω mΩ kΩ MΩ	O MO KO MO
Relation	Percentage	%	%
Relative humidity	Percentage	%	%
Absolute humidity	Gramme/Kilogramme	g/kg	g/kg
Relative change	Percentage	%	%
Frequency	Hertz Kilohertz Megahertz Gigahertz	Hz kHz Mhz GHz	Hz kHz Mhz GHz
Power	Watt Milliwatt Kilowatt Megawatt	W mW kW MW	W mW kW MW
Power (US)	Horsepower	HP	HP
Acceleration	Meter/(s*s)	m/s^2	m/s**2
Torque	Meter/(s*s*s)	m/s^3	m/s**3

NOTE The ASCII exchange format shown in this table is a recommendation for the encoding of the engineering unit if this information needs to be stored or exchanged via a communication network.

Annex F (informative)
UML class diagram semantics

The class diagram is one of the UML specification methods. The UML elements which are used in the class diagrams of this guideline (Figure 5 and Figure 11) are shown in Figure 1.

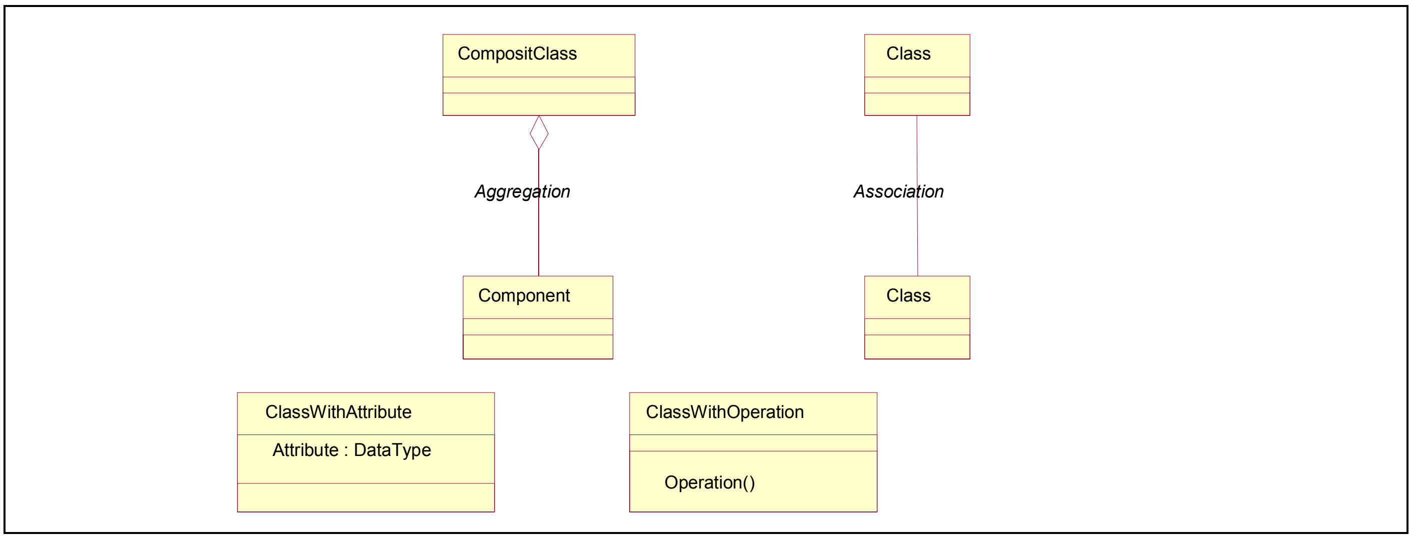


Figure F.1 – Description elements of UML class diagrams

The terms used in the figure are defined in OMG-UML V1.4 as follows:

Aggregation
special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part

Association
semantic relationship between two or more classifiers that specifies connections among their instances

Attribute
feature within a classifier that describes a range of values that instances of the classifier may hold

Class
description of a set of *objects* that share the same *attributes*, *operations*, methods, relationships, and semantics

Component
modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. A component is typically specified by one or more classifiers (e.g. implementation classes) that reside on it, and may be implemented by one or more artefacts (e.g. binary, executable, or script files)

Operation
service that can be requested from an object to effect behaviour. An operation has a signature, which may restrict the actual parameters that are possible

Annex G (informative)
Device classification examples

The following classification should be a support to locate these devices in the entire industrial automation market. The list is based on experiences and good practice of industrial applications. Additionally there are catalogues of devices available in the field of e-business systems. Table G.1 is proposed for use in the first profile development step (see 6.2.3).

NOTE The table is not intended to be complete; however typical field devices of industrial automation are covered.

Table G.1 – Device classification (hierarchy) - examples

Domain/group	Subdomain (family)	Principles (family members)
Power distribution		
	Switchboards	
	Circuit breakers	
	Power monitoring	
	Distribution panel	
Motion control		
	Contactors	
	Protection starters	
	Soft starters	
	Drives	
	Axis control	
	Motor control centre	
	Motor monitoring	
	Positioners	
	Control valves	
Detection, measurement (sensor discrete i/o or analogue)		
	E (electrical)	
	D (density)	
	F (flow)	
		Differential pressure
		Floating body
		Electromagnetic
		Ultrasonic
		Vortex counter
		Deplacement counter
		Turbine wheel counter
		Coriolis
		Thermal
	L (level)	
		Hydrostatic
		Deplacement
		Float
		Ultrasonic
		Microwave
		Laser/optical

		Radiometric
		Capacitance
	Q (quality)	
	P (pressure)	
		Pressure
		Differential pressure
	S (speed, rotary frequency)	
	R (radiation)	
	T (temperature)	
		Resistance, thermocouple
		Pyrometer
		Expansion
		Bimetallic strip
		Hot/cold conductor
	W (weight mass)	
	Distance, position, presence	
		Limit switches
		Inductive sensors
		Photoelectric sensors
		Capacitive sensors
		Ultrasonic sensors
		Pressure switches
Dialogue / operator interfaces		
	Push buttons	
	Joysticks	
	Keypads	
	Pilot lights	
	Stack lights	
	Displays	
	Combined buttons / lights	
	Operator stations	
Logic / universal I/O modules and controllers		
	General input	
	General output	
	Combined input / output	
	Relays	
	Timer	
	Scanners	
	Programmable controller	

Annex H (informative)
Parameter list model

The parameter list model is a simple approach to define a device and consists only of device parameters. A parameter may be made of several sub-parameters that may not be accessible as a single parameter in the parameter list. These parameters are put together as a list where each list entry represents one parameter. The profile template may consist only of a header, a parameter list which can be optional structured in parameter groups and parameter assembly. A graphic device model is not required.

NOTE A simple device may provide the ability to adjust a device's configuration or parameterisation by writing a single value to a specific parameter of the device. No special device behaviour process is provided or necessary.

Annex I (informative) Function block model

I.1 Background

The function block model can be useful for the user of the devices and also the device manufactures as well as the device profile writers. They can apply the function block concept for the design of the device functionality or for designing the interaction between the application control program in the programmable controller and the devices.

Independent of the location of a function blocks (in a device or a programmable controller) its main purpose is the encapsulation of algorithms. The algorithms execute the processing of a set of defined input data and parameters and calculate the desired output data. A single function block or a combination (network) of function blocks builds the control application.

In the design process of a process control application the function blocks are often derived from the *Pipe and Instrumentation Diagram (P&ID)* that is illustrated in Figure I.1 which is well known in the process control industry. The information of the measurement and actuation points together with the designed control structure within the P&ID is transferred to the FB network. In design process of factory automation application function block networks are typically derived from the electrical design schemes (e.g. ECAD).

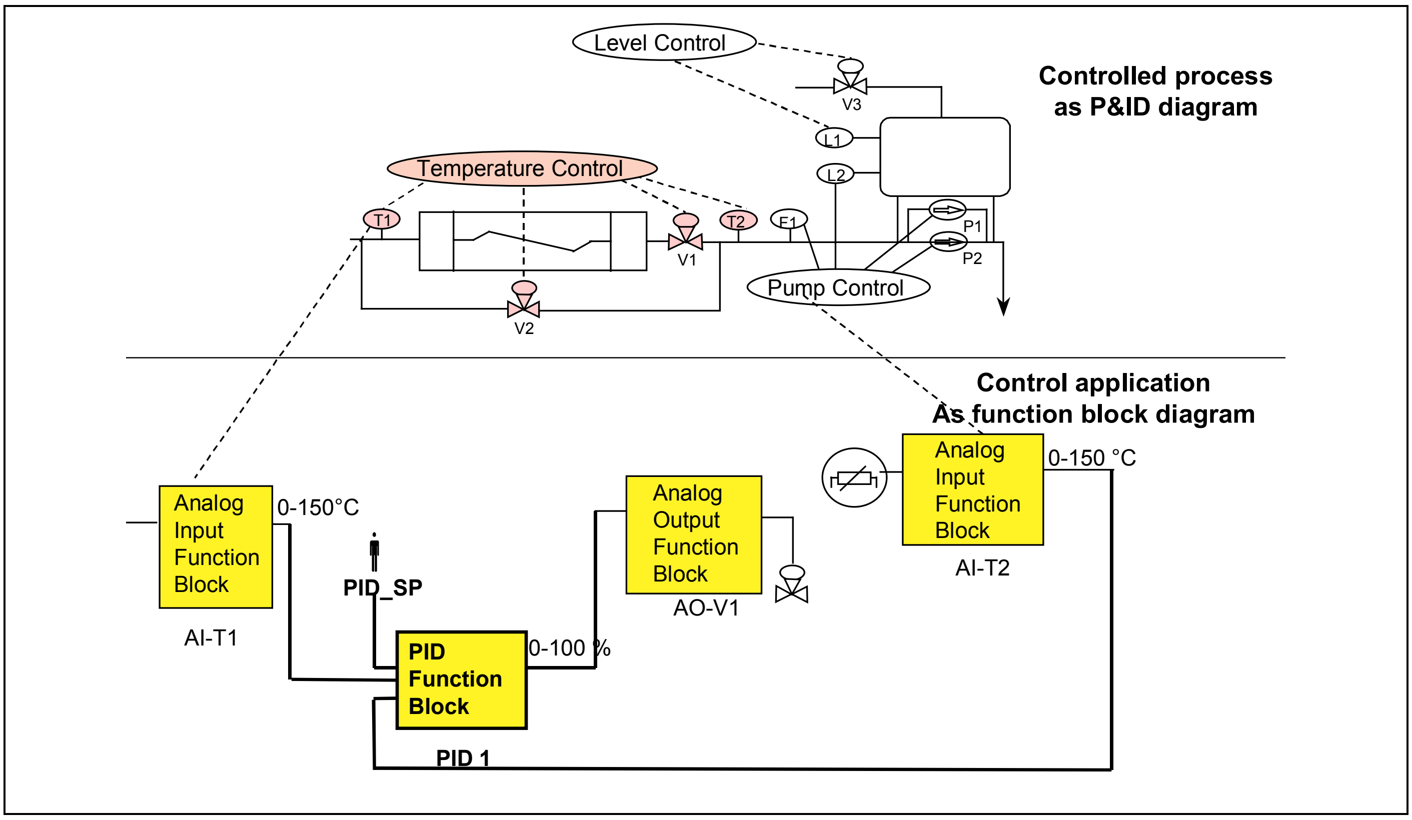

Level Control
V3
Controlled process as P&ID diagram
L1
L2
Temperature Control
T1
V1
T2
F1
P1
P2
Pump Control
V2
Control application As function block diagram
Analog Input Function Block
0-150 °C
AI-T2
Analog Output Function Block
AO-V1
Analog Input Function Block
0-150°C
AI-T1
PID_SP
PID Function Block
0-100 %
PID 1

Figure I.1 – Function block diagram derived of the P&ID

Figure I.2 illustrates that the function blocks can reside in the field devices (FD), in the programmable controllers and also in the visualisation tools.

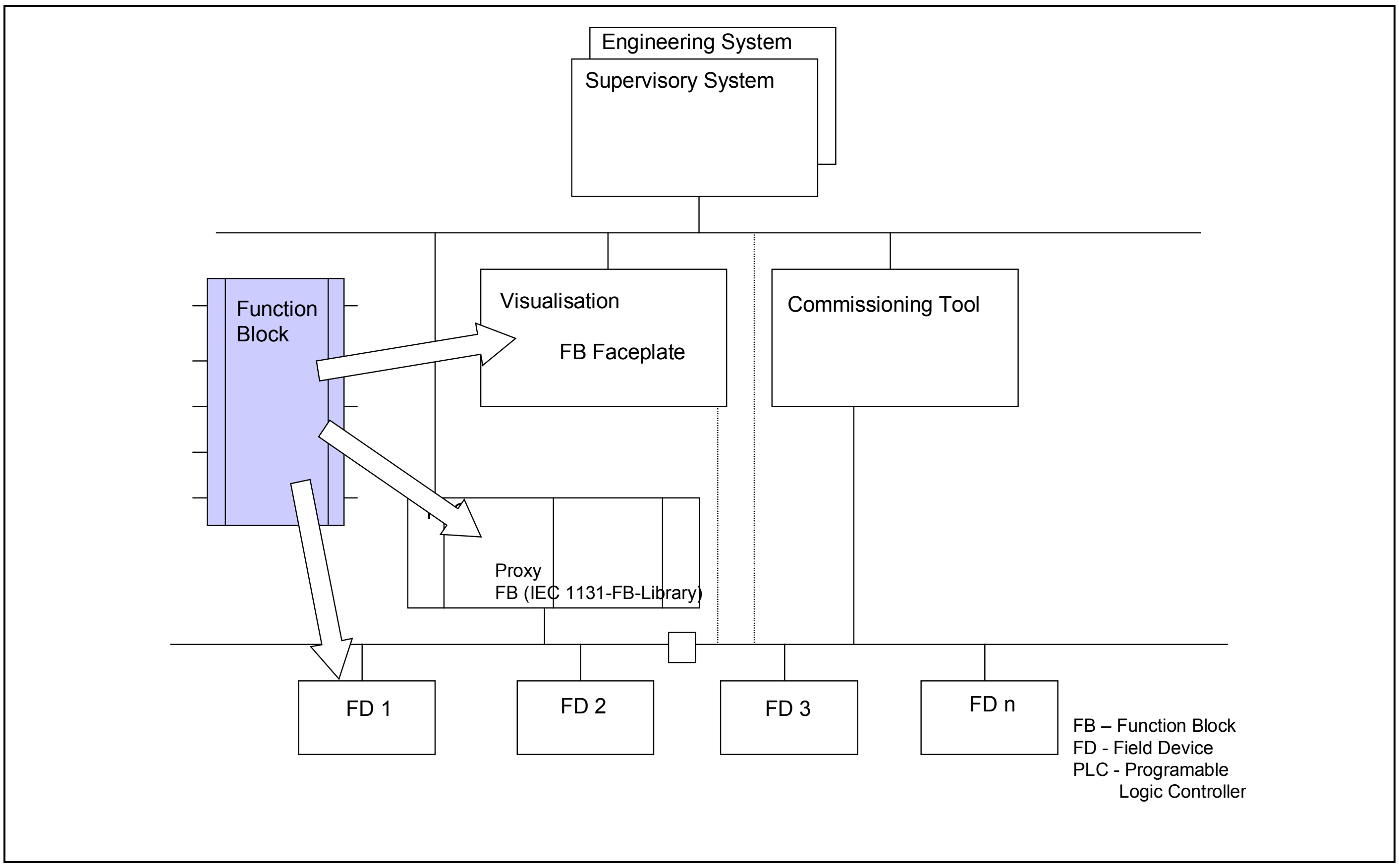


Figure I.2 – Function blocks implemented in different devices

I.2 Control system structure paradigms

There are three main paradigms dealing with control system structure using function blocks as shown in Figure I.3.

The classical system structure is the *central control* with a function block implementation typically written in IEC 61131-3 languages using central and/or remote I/O modules e.g. on a field bus. The typical modules are simple devices or devices with a interface like simple devices. To address the I/O data the application program uses the directly represented I/O variables which are defined in IEC 61131-3 as e.g. I1.1 or Q2.3.

An increasing number of systems make use of the *decentral control* structure with intelligent field devices connected by a field bus. Here the proxy function block concept applies specific function blocks in the "central" PLC application program which represent the functionality of the devices and realize the implicit communication to the device over the field bus. In this case the devices are addressed by the input and output variables of the proxy function blocks. The application program with the function blocks in the controller is programmed "centrally" scheduled according IEC 61131-3 or IEC 61804.

A new paradigm is the *distributed control* structure with function blocks distributed over networks which communicate directly as defined in IEC 61499. In this case the application program is distributed in the devices and the controllers (if any). In contrast to the proxy concept of the decentral structure here the function blocks need a system wide scheduling or a event driven mechanism for their invocation.

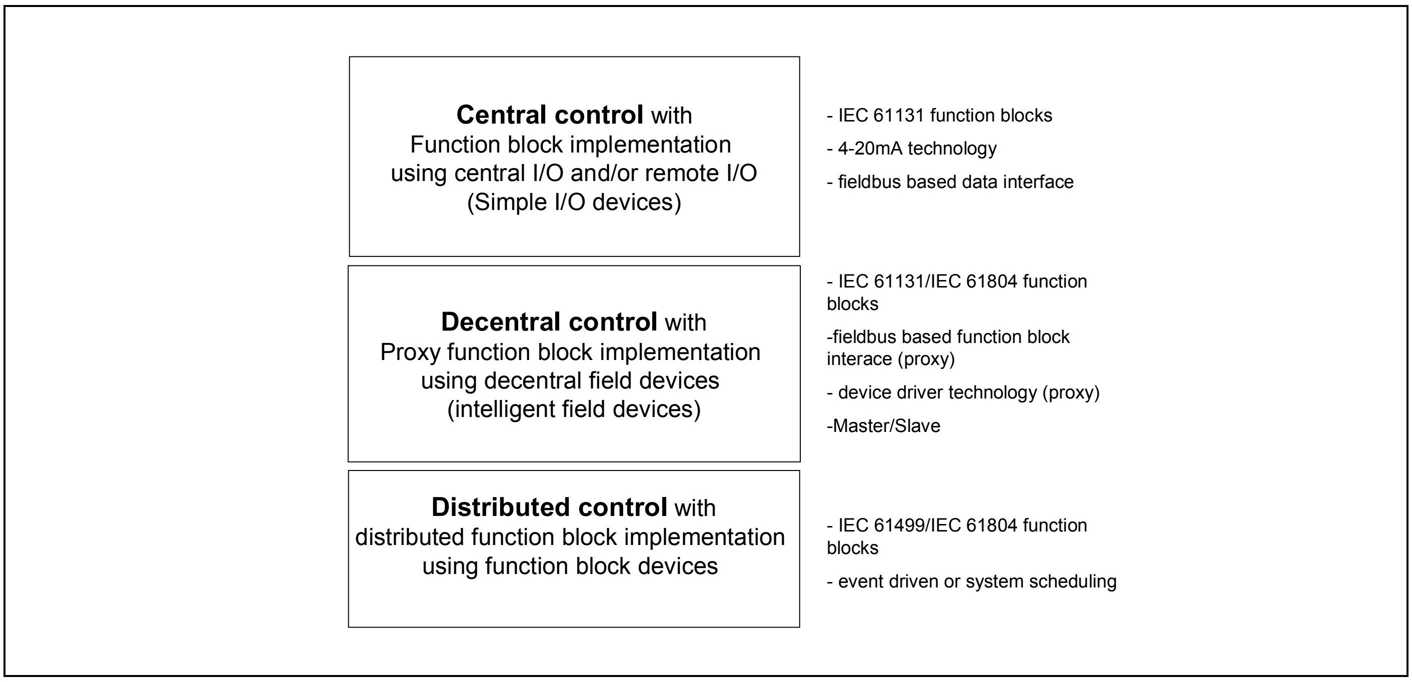

Central control with
Function block implementation
using central I/O and/or remote I/O
(Simple I/O devices)
- IEC 61131 function blocks
- 4-20mA technology
- fieldbus based data interface
Decentral control with
Proxy function block implementation
using decentral field devices
(intelligent field devices)
- IEC 61131/IEC 61804 function blocks
-fieldbus based function block interace (proxy)
- device driver technology (proxy)
-Master/Slave
Distributed control with
distributed function block implementation
using function block devices
- IEC 61499/IEC 61804 function blocks
- event driven or system scheduling

Figure I.3 – Function block application in control system structure paradigms

I.3 Function block paradigms

The basic function block model which is common for all IEC 6113-3 languages and also used in IEC 61499 and IEC 61804 supports the following two paradigms:

It reflects the *component* paradigm of the reusable logic modules derived from the hardware design method with defined interface and encapsulated functionality including a persistent data memory.

It also includes the *type/instance* paradigm of the object oriented IT programming method. That means a piece of program (algorithms), which is supposed to be reusable is to be programmed as a function block type and stored in a library. A function block type consists of a declaration and a body as shown in Figure I.4. In the application program this function block type can be invoked (called) as one ore more function block instances with different parameter sets. In the context with this device profile guideline each instance can represent a device with local memory data.

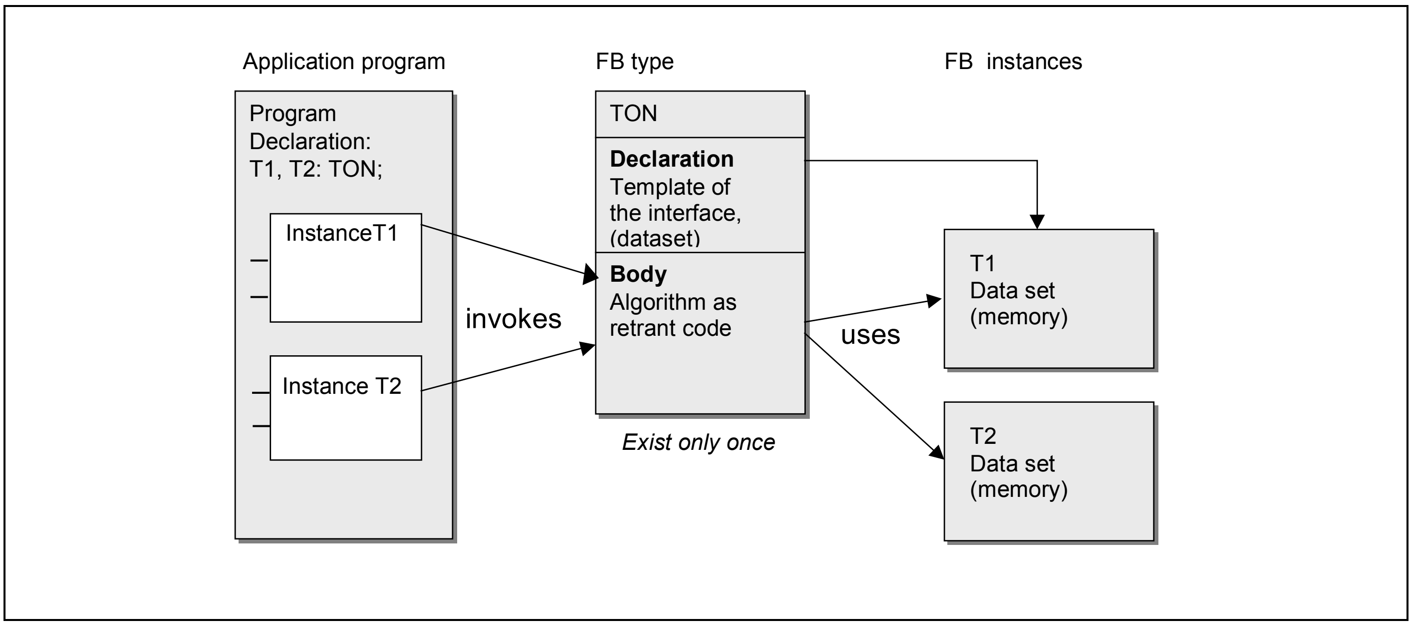


Figure I.4 – Function blocks of IEC 61131-3

In IEC 61131-3 mainly the principle of a central control with direct data access is considered even if the data is accessed over a field bus as shown in Figure I.5.

I.4 Proxy function block

Figure I.5a) shows a motor control application consisting in a function block and some combined basic blocks (AND, OR) that directly execute the process I/O. In the process I/O the motor terminals and some associated interlocking signal are mapped.

In Figure I.5b) the above mentioned “combined basic blocks” are hidden in a predefined function block specific motor control that represents the device and hides the communication. For the user of this motor control function block is not visible whether there is central I/O or remote I/O connected. This usage of function block is called *proxy function blocks.*

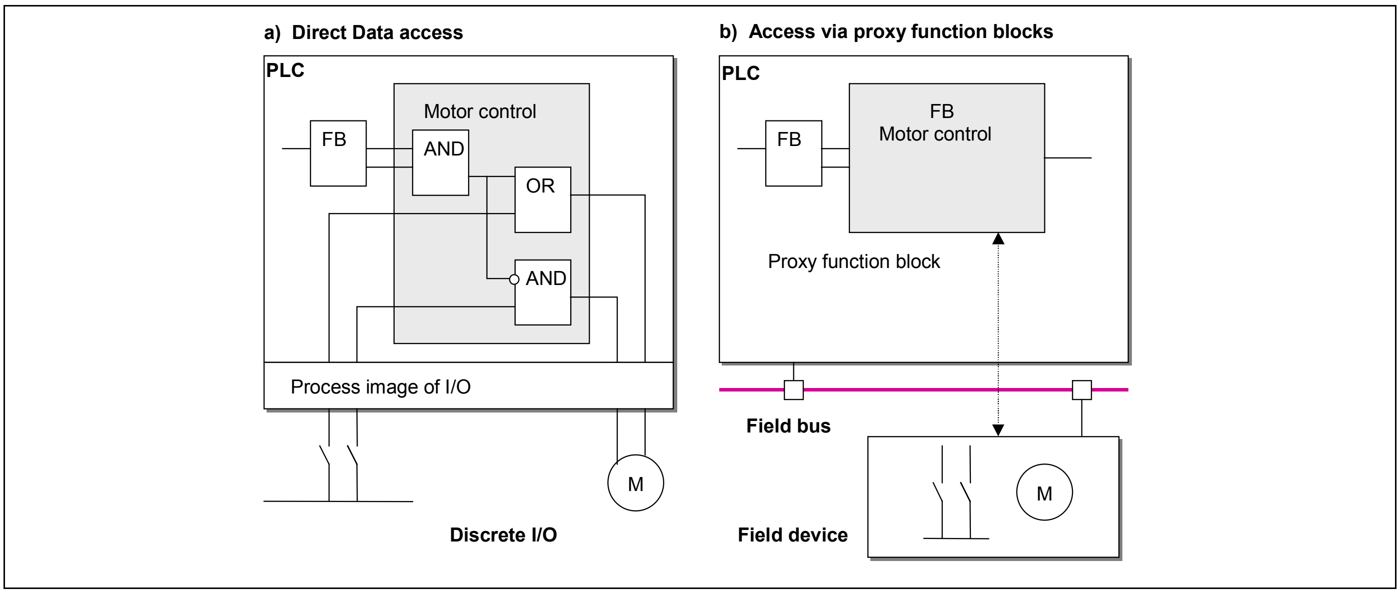


Figure I.5 – Concept of a central controller according IEC 61131-3

There are various possibilities in a PLC program to access to the data in remote modules and devices:

A typical solution in the PLC is the "cyclic access" via the so-called *process image* to the remote inputs and outputs. In the application program of the PLC these remote variables are used like local I/O variables. The data exchange over the fieldbus is cyclically executed. The variables are transferred independently of the execution of the application program and mapped in the process image.

To achieve a standardized communication between the PLC and the device a set of *Communication Function Blocks* can be offered by the system manufactures. Reasons for this standardisation are:

Standardised communication function blocks are the basis for portable PLC programs

Different process states usually need different sets of device parameters which have to be transferred. Therefore selective parameter access provide a better communication performance

Figure I.6 shows a PLC as a fieldbus master and a device as a fieldbus slave. They communicate by using generic communication function blocks e.g. read data record and write data record which can be used by application programmer.

Figure I.6 shows also an instance of the *Proxy Function Block* representing the field device in the IEC 61131-3 application program in the PLC. This device specific *Proxy FB* exhibits the functionality of the device by its input and output parameters. Inside the Proxy FB basic *Communication Function Blocks* provide the reading and writing access to the field device data using the standard fieldbus protocols.

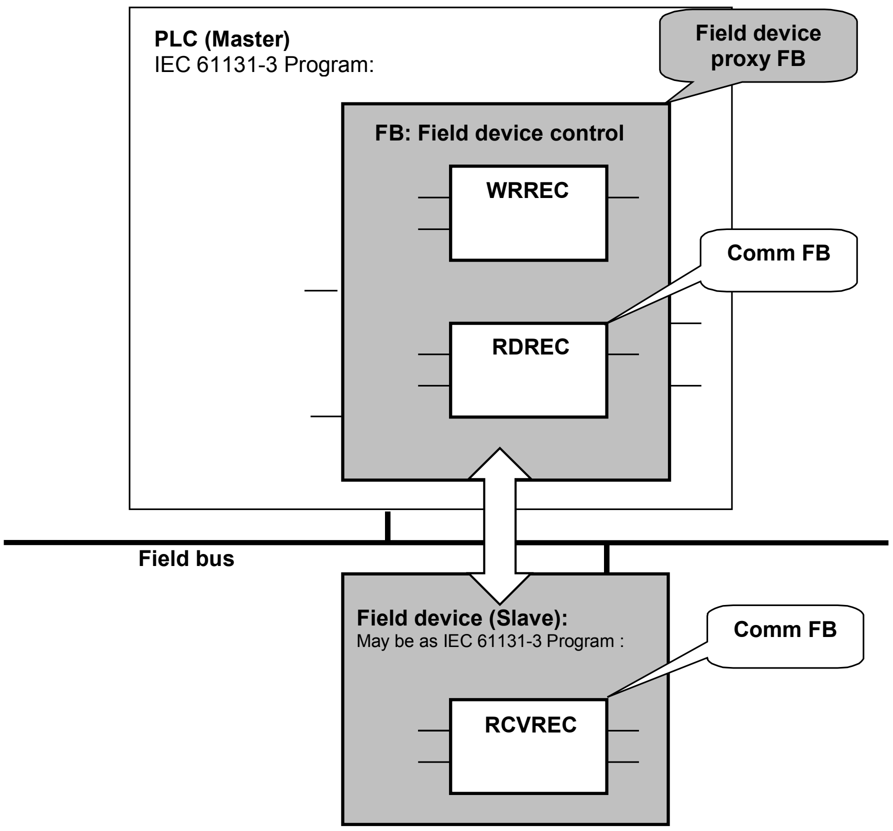


Figure I.6 – Proxy FB and Communication FB

In smart devices it is possible to describe the device functions in terms of FBs. This is the basic idea of IEC 61804. The main reason is, that the application design may integrate the decentralised field device functions in terms of FB in the entire application. One typical implementation variant uses the proxy concept. The PLC FB network integrate the field device FBs in its data flow, the communication between controller and field device is hidden in the proxy using the communication FBs (see Figure I.6).

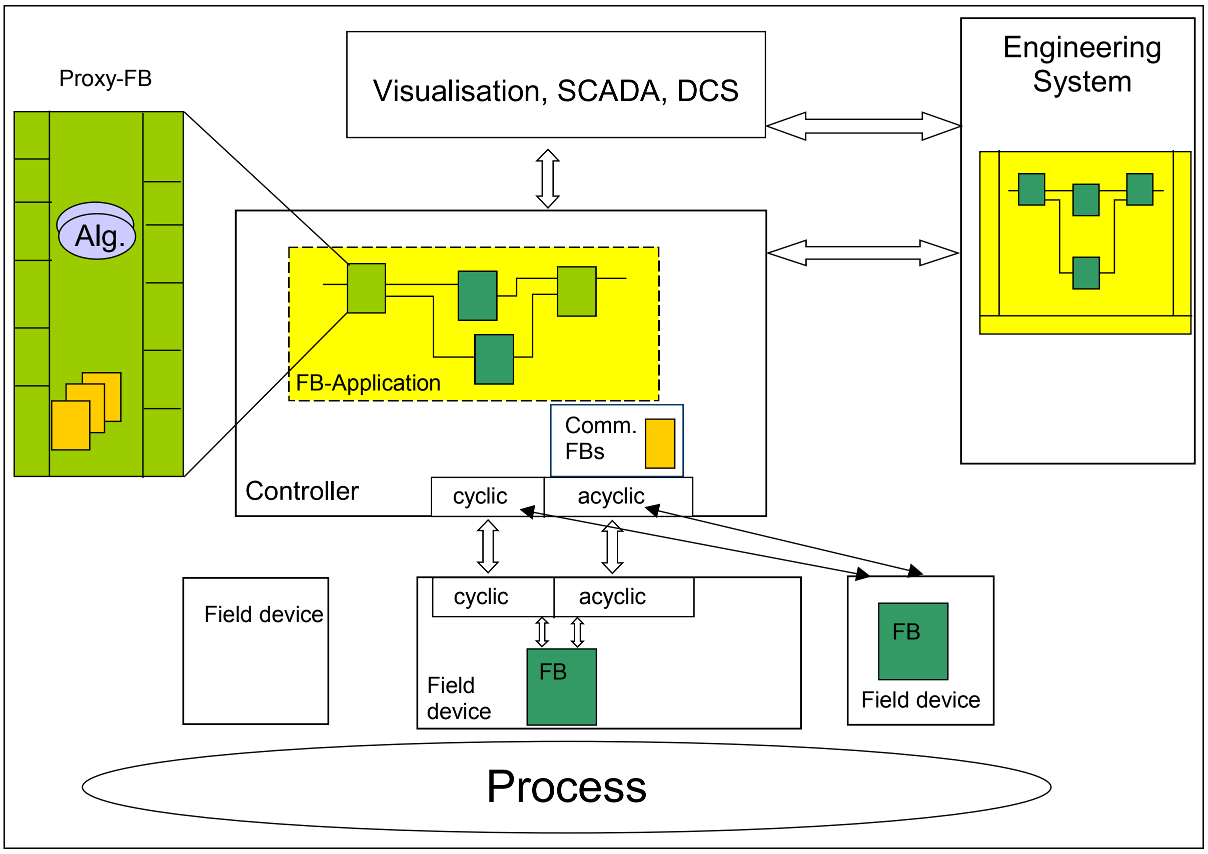


Figure I.7 –Function blocks in field devices and their integration in control programs

It is possible to design data flow between FBs in field devices without their integration in the controller program if additional scheduling and management functionality are implemented in the devices.

I.5 Distributed control structure according IEC 61499

The distributed control system structure paradigm according IEC 61499 is illustrated in Figure I.8.

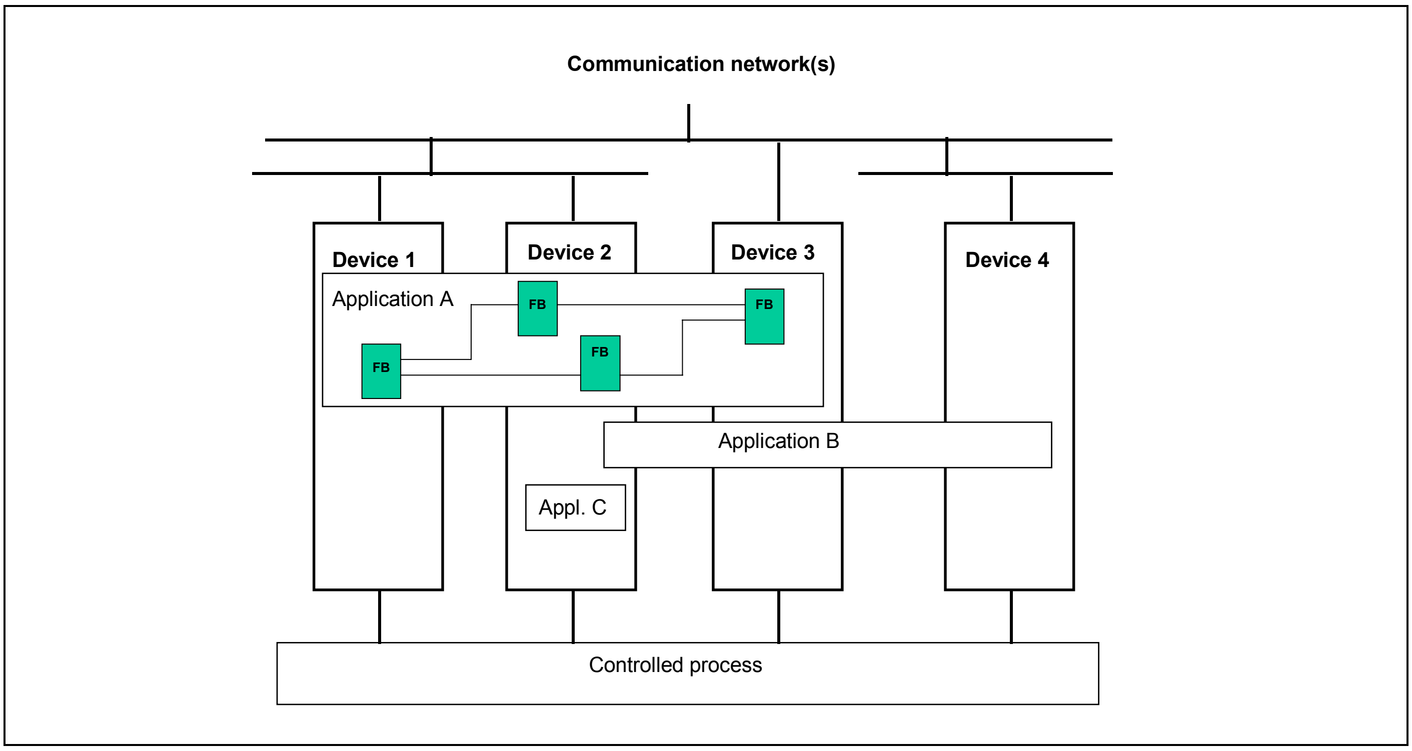


Figure I.8 – Function blocks applications distributed in devices according to IEC 61499

According IEC 61499 the entire functionality is located in field devices using distributed function blocks. Since here the scheduling of the function blocks by the common operation system, e.g. of the PLC is missing the invocation of the function blocks is done by the events coming from other function blocks. The data flow can be coupled with the event flow from function block to function block as shown in Figure I.9.

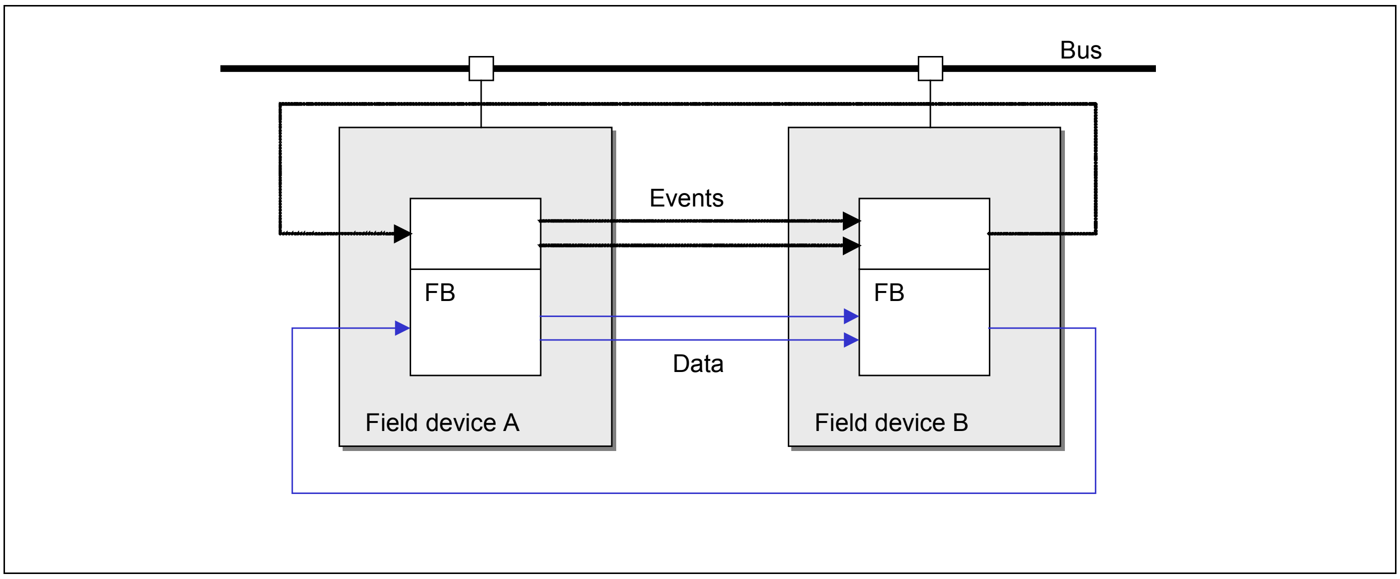


Figure I.9 – Application program distributed among field devics

I.6 Function block model in the process control devices

In the process control industry often the function block oriented device model is used as shown in Figure I.10. The figure shows the *functional elements* of a device modelled as function blocks [IEC 61804-2].

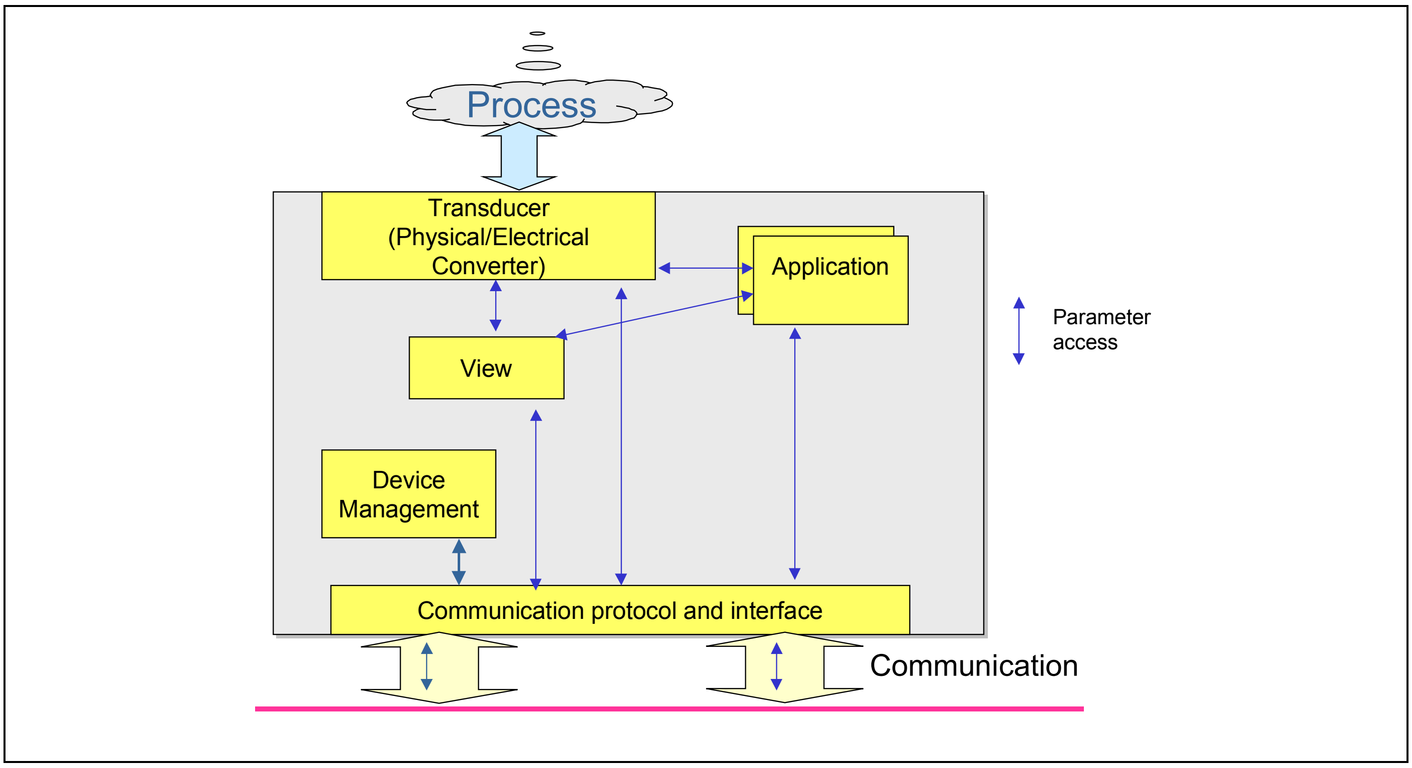

Process
Transducer (Physical/Electrical Converter)
Application
View
Device Management
Communication protocol and interface
Parameter access
Communication

Figure I.10 – Function block device model in a field device

NOTE The communication profile part is not in the scope of this guideline

The following list gives explanations for the function block types of Figure I.10:

Device Management function block

Identification information (boiler plate), diagnosis, device controlling state machine

Transducer function block

Process attachment control

Application function block

Process control application dependent signal processing, e.g. manual/automatic operation modes, limit checking, scaling

View

Concatenation of function block parameters for communication optimised parameter access

Annex J (informative) Object model

J.1 Background

The use of object modelling within automation applications has been triggered by increasing requirements from device manufacturers, system designers and users for:

easier integration of plant databases

improved interoperability

reuse of existing application knowledge for design, operation and maintenance.

In an industrial environment where public and private networks are more and more often an integral part of the overall enterprise architecture, plant-floor devices may be required to interoperate with corporate information applications, in addition to support their usual control interface. Besides, users are requiring that devices from different vendors interoperate in the same application.

In parallel, control systems paradigms have gradually evolved:

In the centralized control system structure, a single controller, connected to I/O modules or similar simple devices, handles the complete application and data exchanges

In the decentralized control system structure, controllers may be additionally connected to intelligent devices with local processing capabilities, but will still handle most data exchanges

In the distributed control system structure, both controllers and devices handle their part of the overall application and exchange data as needed, using peer-to-peer communication

The object approach may be used for all three paradigms, directly in the case of distributed control, or via the definition of mapping/interface objects in the other cases.

Object modelling helps provide:

Consistent characteristics for all devices

A well defined externally visible behaviour

Common interfaces and services

Uniform description of exchanged information

The definition of a library of re-usable objects and device profiles based on these objects reduces the design time and facilitates plug-and-play interoperability between complex devices from different manufacturers.

J.2 Object modelling concepts paradigms

J.2.1 Object elements

An object provides an abstract representation of a particular component within a device.

Traditional application design included data structure, functions and processes: object modelling organizes related data and procedures into a single entity, using a specific terminology, and refers to it as an object (see Figure J.1).

NOTE The actual mapping of the abstract object model within a device is implementation dependent.

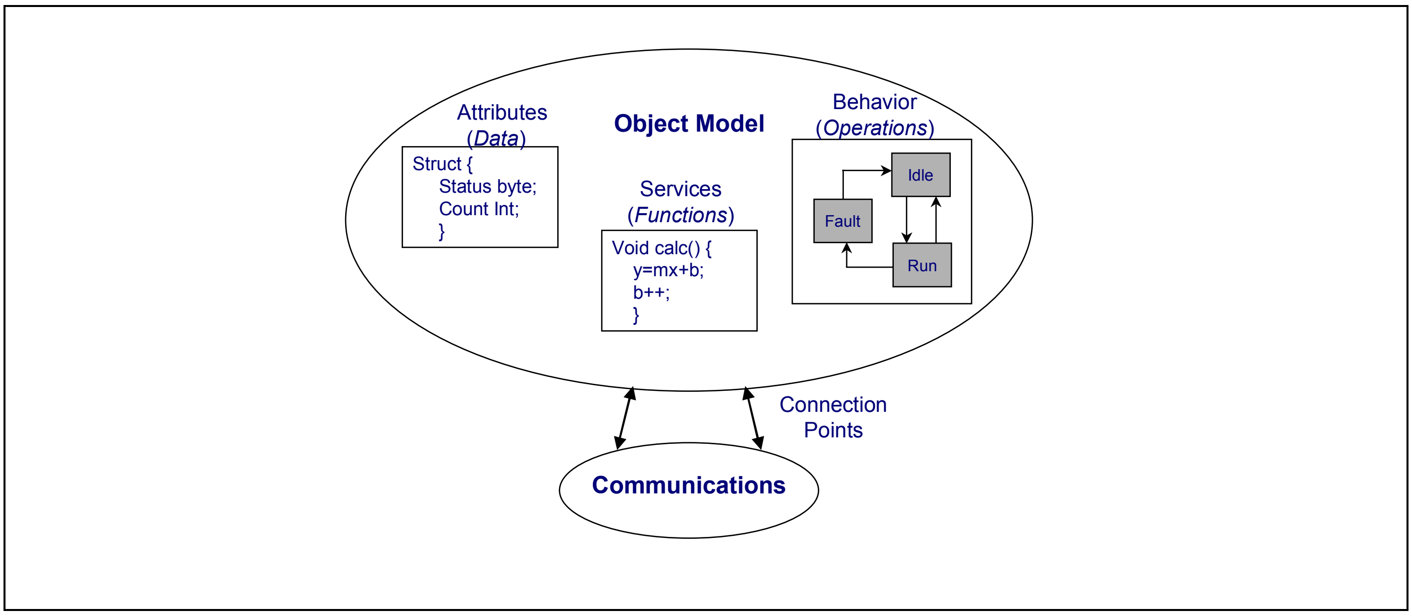


Figure J.1 – Comparison of object model with traditional design

An object is composed of the following elements:

a set of related attributes (data);

services (functions);

defined behaviours (algorithms);

supported connection points (for mapping onto communication).

Attributes are represented by values or variables, and provide a description of externally visible characteristics or features of an object.

Attributes may be used to exchange information about the process signal flow, e.g. to provide status information, or to govern the operation of an object: the behaviour of an object may be affected by the value associated with an attribute.

A service is invoked to trigger an object to perform a task: it provides a function supported by the object. Object modelling supports both common (e.g. get, set, reset) and object-specific services. Object-specific services are those that are defined for a particular object class to perform a required function not covered by a common service.

Services may be used to provide the following functions:

Access to various object attributes (get and set), e.g. to provide input for process signal flow (set point) or to provide a parameter for scaling functions

Trigger transition of the internal state machine governing object behaviour (e.g. start, stop, resume)

Start specific operation/algorithm featured by the object

Retrieve information about dynamic object structure or interface

The behaviour of an object indicates how it responds to particular events. Actions result from different events of which the object is aware, such as receiving service requests, detecting internal faults, elapsing timers, or a change in an attribute value.

Connection points are object interfaces with the communication system.

J.4 Object classes

A class is a group of objects that defines a particular type of object and defines the characteristics shared by all the objects within a class.

Objects within a class are called object instances. An object instance is the actual representation of a particular object within a class. Each instance of a class has the same behaviour and the same set of attributes, but has its own set of attribute values, which makes each instance in the class unique.

EXAMPLE The class “human” may be characterized by a common set of attributes (e.g. gender, age, size, eye colour). Every person can be associated with an instance of this class, with a different set of values for the attributes, which allow individual persons to be distinguished.

Multiple object instances within a particular class can reside in a device. Object items can be uniquely identified within a device by using a class identifier, an instance identifier and an attribute identifier, as needed (see Figure J.2).

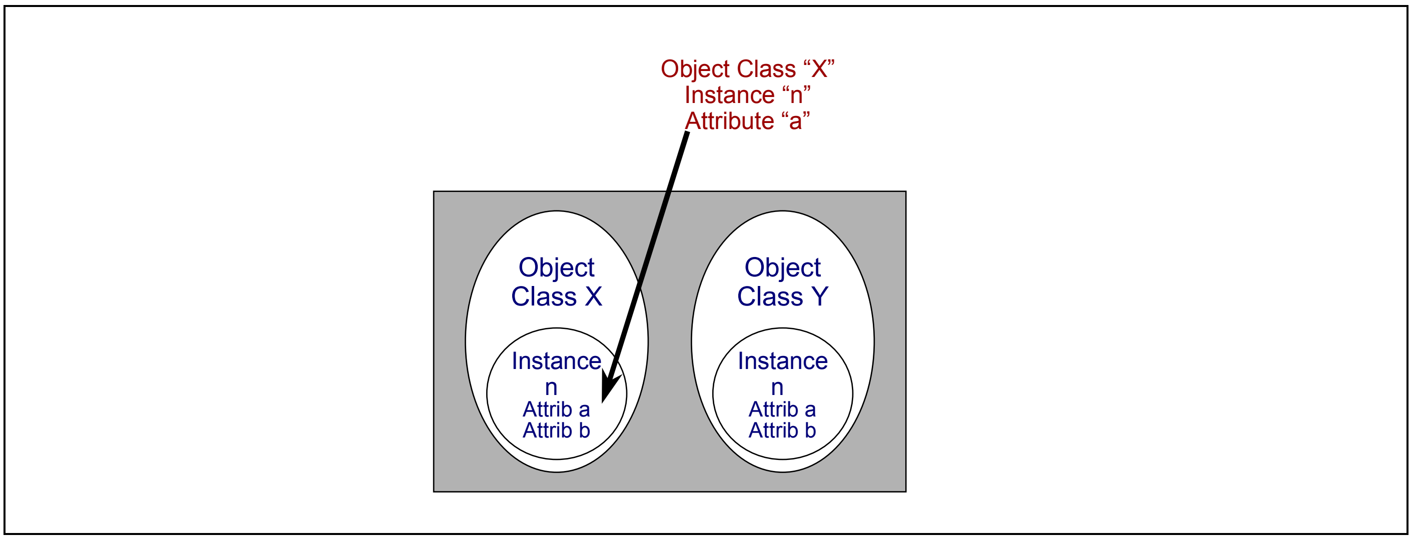


Figure J.2 –Object addressing

J.4 Device object model

J.4.1 Overview

A device may be modelled as a collection of objects. Figure J.3 shows the basic device object model, with the corresponding mapping onto the classes defined in ISO 15745-1.

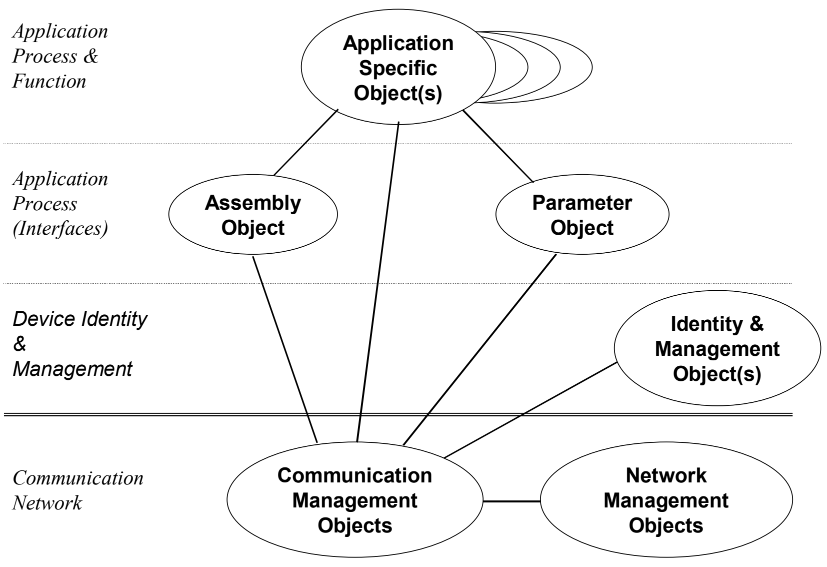

Application Process & Function
Application Specific Object(s)
Application Process (Interfaces)
Assembly Object
Parameter Object
Device Identity & Management
Identity & Management Object(s)
Communication Network
Communication Management Objects
Network Management Objects

Figure J.3 – Device Object Model

J.4.2 Object categories

The Identity and Management object(s) contains the characteristics of the virtual device that are independent of the network being used and the process being controlled, such as manufacturer identification, part number, revision. It also provides device status information, and supports services to control the device, i.e. device administration.

Application specific objects are designed for a particular type of product or product range. They are mostly used to describe the behaviour of the device in terms of the application (e.g. signal processing). Some may also be used to describe the intrinsic function of a device in terms of its technology, if there is a need to exchange information related to this device technology (e.g. specific calibration).

Assembly and Parameter objects are general purpose application objects. They are intended as interfaces between the Application objects and the communication system.

Assembly objects allow an optimised access to object attributes by gathering attributes from several objects into a single object: they are typically used for I/O exchanges (see Figure J.4).

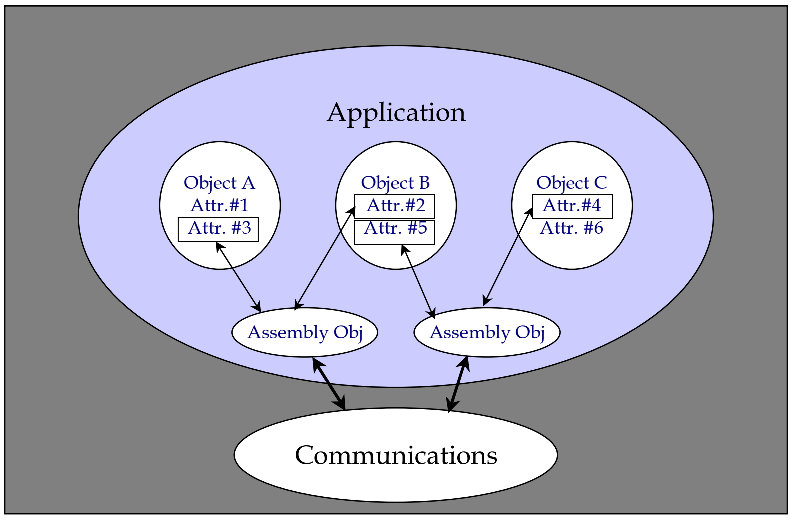


Figure J.4 – Assembly object

Parameter objects are used to provide a common method to access device data (i.e. typically attributes of different application objects, but not necessarily) : they contain parameter characteristics, i.e. information needed to interpret or display this data (see Figure J.5 and Annex B).

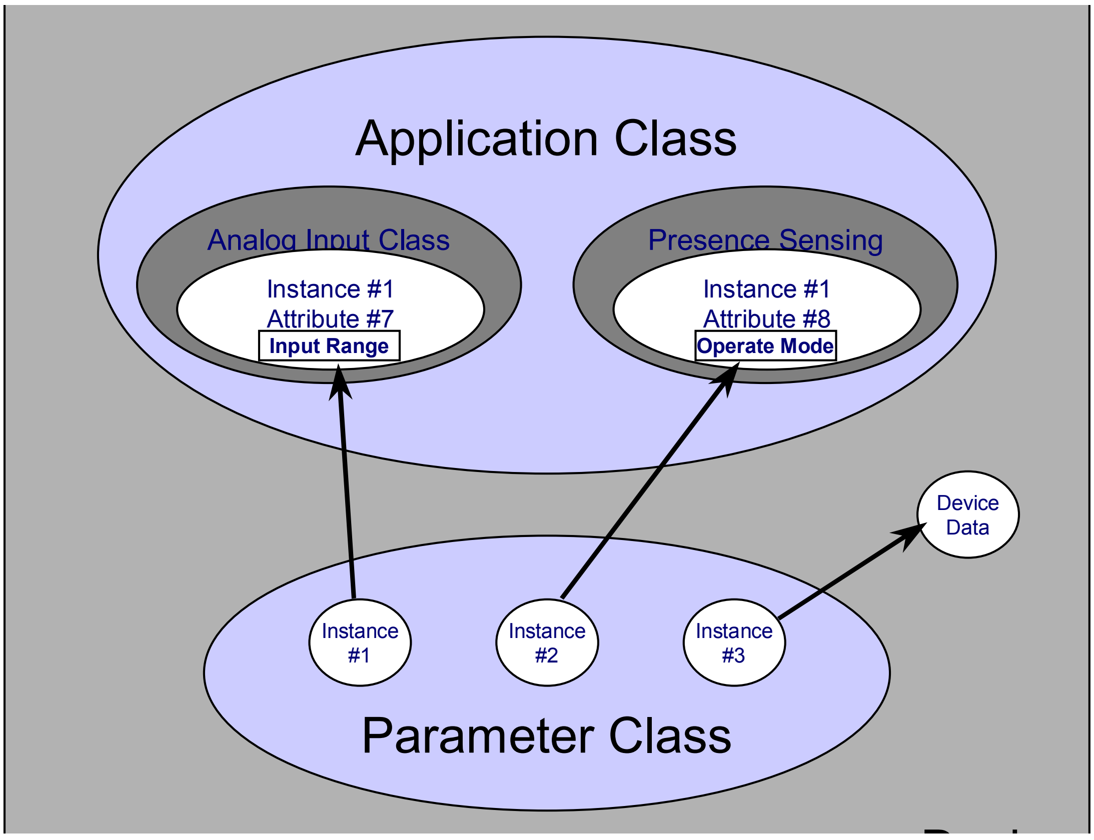


Figure J.5 – Parameter object

NOTE These two types of objects are typically used in centralized or decentralized control systems.

Communication/network management objects are used as interfaces between the device application objects and the network, or for network management (e.g. setting and monitoring of data link layers parameters). Figure J.6 shows an example on how communication management objects may be used for various types of data exchange (e.g. I/O data or configuration and diagnostics).

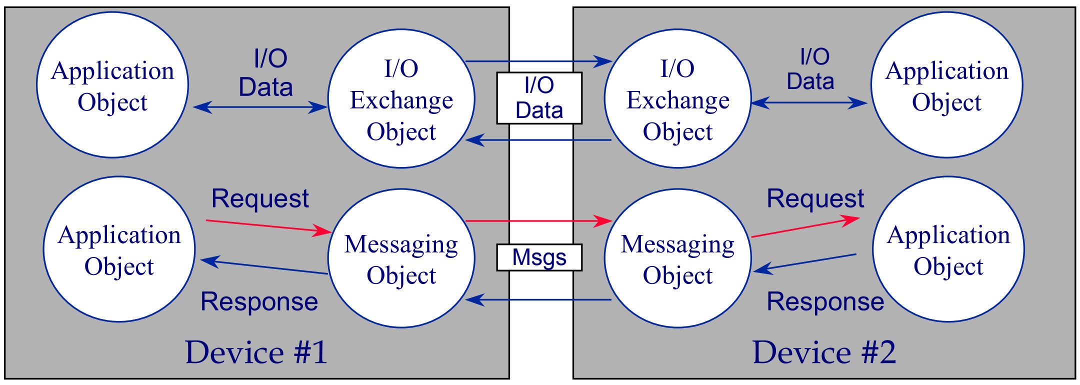


Figure J.6 – Communication management objects example

BIBLIOGRAPHY

IEC/FDIS 61158-5:2003, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems - Part 5: Application Layer Service Specification*

IEC/FDIS 61784-1:2003, *Digital data communications for measurement and control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*

IEC 61850 (all parts), *Communication networks and systems in substations*

IEC 61987 (all parts), *Industrial-process measurement and control – Data structures and elements in process equipment catalogues*

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*

IEEE Std 754-1985 (R1990), *IEEE Standard for Binary Floating Point Arithmetic*

REC-xml-20001006, *Extensible Markup Language (XML) 1.0 Second Edition – W3C Recommendation 6 October 2000*

REC-xmlschema-1-20010502, *XML Schema Part 1: Structures – W3C Recommendation 02 May 2001*

REC-xmlschema-2-20010502, *XML Schema Part 2: Datatypes – W3C Recommendation 02 May 2001*

UML V1.4, *OMG - Unified Modeling Language Specification (Version 1.4, September 2001)*