65/290/DC

2002-03-29

INTERNATIONAL ELECTROTECHNICAL COMMISSION

TC65: INDUSTRIAL PROCESS MEASUREMENT AND CONTROL

Circulation of Draft for Comments (DC) prepared by PJWG on:

Device Profile Guideline

Comments shall be sent using the IEC compilation frame to TC65 Secretary

before:

June 28, 2002 at the following email address: **iec@dumortier.org**

This document is also to be circulated for comments to NC's of TC's or SC's:

SC 17B, SC 22G, TC 57, SC 65A, SC 65B, SC 65C and ISO TC 184/SC 5.

Secretary background explanation note:

Device profiles specifying the device behavior, parameters, and configuration data, are currently described differently in technical committees dealing with e.g. switch gears (SC17B), drives (SC22G), power substation (TC57), process (TC65), and manufacturing automation equipment (ISO TC184). At its meeting on January 11, 2000, the Sector Board 3 identified an urgent need to align the various approaches in order to facilitate systems integration and save engineering costs, and encouraged the committees to constitute a group of experts.
TC 65 offered [65/251/INF] the opportunity of its Advisory Group meeting on February 4, 2000 to discuss the setting up of the task force and identified the scope of the study.

The tasks were:

1. to analyze the different methods currently used in IEC SC 17B, SC 22G, TC 57, TC 65 and its SC's, and ISO 184 projects;
2. to locate the common objectives;
3. to identify common principles, content requirements (inclusions; exclusions), terminology;
4. to find a common view on how to write a device profile;
5. to propose if a generic outline, or guideline, or standard for device committees is useful; and

6. to propose next actions.

The Task Force convened by Mr Joe DUFFY, Chairman of SC 65A, made the following recommendations to Sector Board 3 and its committees:

1. A joint working group should rapidly develop a Guideline taking into account the ISO 15745 approach of the Device Profile Class Diagram. This guideline is intended for TC's developing standards for how to develop profiles (e.g. SC17B) and for TC's developing profiles (e.g. TC57).
2. The joint working group will develop a generic device reference model, which supports some standard basic capability and allows device manufacturers to enhance the capability of the device. The Profile guideline will be based on this generic device reference model. The Profile Guideline should support description of both mandatory and optional capabilities.
3. The joint working group developing the profile guideline and the generic device reference model should take into account the NOAH project work along with the ongoing work in the Technical Committees.
4. Each technical Committee should review the current status and priority of its present project and determine when and how to start using the guideline and generic device reference model. Each TC needs to look at the availability target for the Guideline and generic device reference model and determine how it will impact the current work of the technical committee. The primary focus of the recommendation from this task force is to impact new work after the guideline and the generic device reference model is available.
5. Each Technical Committee on this task force that is currently developing a profile has identified some areas of potential overlap with one or more other TCs. SC17B & TC57 for example. These TCs should identify these potential areas of overlap and coordinate the profile development immediately.
6. Each TC presentation used similar terminology and had similar but not identical meanings for this terminology. The Guideline should make precise definition of terms.
7. The IEC/ISO organizations should decide on the appropriate type of publication for the Guideline which should be available within 2 years.

After successful start of the joint Task Force, made of representatives of TC's or SC's identified above, Mr. DUFFY asked to resign from the convenorship given to him at the Nice AG meeting. Mr. Hans-Peter OTTO was proposed by the Task Force members as the new convenor, and following the recommendation of the SB3 Secretary the Task Force should have the new name: Preliminary Joint Working Group (PJWG) on Device Profiles. Moreover, it was discussed that it would be more appropriate to have this PJWG Device Profile under the leadership of TC65 since it appears that the resulting work (Guideline for profile writers) will be of interest of several TC's of IEC and ISO. [65/270/INF]

Presently, a draft document prepared by PJWG is available. Hereafter, this draft for comments (DC) is attached and circulated for comments in National Committees.

Secretary note:

A full report on PJWG Device Profile is at the agenda item #9 of TC65 meeting to be held in Beijing on April 12, 2002 to consider further steps. Possible circulation of this specification as a PAS, taking into account comments provided on this DC, will be discussed during the next TC65 meeting to be held in Beijing

Bernard Dumortier
TC 65 Secretary

Device Profile Guideline

Document for Comments (DC)

Editors note: This Document for Comments is prepared by the IEC TC65 Preliminary Joint Working Group, PJWG) installed by the IEC/ISO Sector Board 3 “Industrial Automation”. The PJWG consists in experts of the following committees: IEC 65, IEC 22G, IEC 17B, ISO 184/SC5, IEC 57.

Table of contents **page**

Table of figures

Table of tables

Introduction

This guideline is a recommended outline for use by IEC Product Committees and product manufacturers to develop and provide profiles for networked 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 industrial devices. This variation makes determining device interoperability, interchangeability, comparisons and common device behaviour more difficult. Therefore it is the intent of this guideline to provide a common and more generic way to convey device information and behaviour to reduce the present variation and to reduce the total cost of the industrial control system.

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

It is the further intended of this guideline to provide recommendations for conveying in an electronic file the necessary device information to non-human users of the device profile such as software tools and application programs. These recommendation includes the use of standards such as XML (eXtensible Markup language).

1 Scope

The scope of this guideline will allow device profiles to be developed for the entire range of industrial field and control devices from simple to complex. The construction of simple device profiles such as limit switches and contactors for simple bit level device networks, via medium complex devices such as transmitters and actuators for process control to complex device for field buses such as variable speed drive shall be supported by this guideline.

This guideline is not intended for industrial processor controllers such as Personal Computers and Programmable Controllers. However, it is recognized that some devices may contain characteristics and abilities similar to these controller. A device may be capable of carrying out logic, timing and counting functions such as AND, OR, NOR, ON Delay, OFF Delay, etc. The guideline will allow device to be constructed as device made up of slots and/or modules.

This guideline recognizes that a complete device profile may consist of multiple parts. One complete representation could consist in the physical and functional components of the devices and also of their communication and application aspects. Each of these components has an intended purpose to convey a particular aspect of the device to the users of the profile. This guideline recognizes that each profile part may be useful standalone or for various different users.

Further this guideline presents a device model to better guide and delineate a device profile’s content. The guideline will allow device profile’s content to provide different aspects of the meta model.

The device model allows the use of function blocks and/or object oriented techniques to convey the structure and behaviour of the device model in a unique manner.

This guideline will also allow for the device and its profile to contain the ability to carry out complex internal operations such as calibration, scaling, programming.

To be useful to users a common method for conveying the device profile information is required. This guideline will recommend the use of uniform device profile templates. This will allow users of these profiles to make comparisons, determine interoperability and interchangeability, and recognize common device behaviour.

Since some industrial devices can also carry out portions of the industrial control application in their microprocessor controller this guideline will allow device profiles to contain a means for a device to be programmed with appropriate logic and/or computing functions.

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

IEC 61131:1994, Programmable controllers - Part 3: Programming languages

IEC 61158:2000, Digital data communications for measurement and control – Fieldbus for use in industrial control systems - Part 5: Application Layer Service Specification

IEC PAS 61499:2000, Function blocks for industrial process measurement & control - Part 1: Architecture, Part 2: Technical specification

IEC 61784:2001, Communication profiles for field bus – Profile sets for continuous and discrete manufacturing

IEC CD/Draft 61804:xxxx, Function blocks for process control - Part 1: Generic requirements, Part 2: Specification

IEC CD 61850:xxxx, Communication networks & systems in substations

IEC CD 61915:xxxx, Low-voltage switch gear and control gear – Profiles of networked industrial devices

ISO CD(V) 15745:xxxx, Open systems application integration frameworks - Part 1: Generic reference description.

ISO/IEC 19501-1: Information technology -- Unified Modelling Language – Part 1 Specification (UML)

3 Definitions

The following list contains the terms and their definitions applicable for the common device model and guidelines.

[Editors note: The list of definition is not yet in fully line with the document. It will be up dated in the next draft revision]

3.1
algorithm
completely determined finite sequence of instructions by which the values of the output variables can be calculated from the values of the input variables [IEC Dictionary 351-11-21]

3.2
application
software *functional unit* 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.

3.4
class (of a device)
description of a set of *objects* that share the same attributes, operations, *methods*, relationships, and semantics [ISO 19501]

3.3
class attribute
feature within a classifier that describes a range of values that *instances* of the classifier may hold [UML]

NOTE It is a property of a class instance (object).

3.5
data
collection of factual information (as measurements, statistics, or set of parameter values) digitally transmitted in numerical form that can be used as a basis for reasoning, discussion, or calculation

3.6
data dictionary
complete listing of all I/O-data, *parameters* and their parameter attributes of a *device*

3.7
data type
set of values together with a set of permitted *operations* [? 15.04.01]

3.8
(field) device
networked piece of equipment of an industrial automation system capable of performing specified functions in a particular context and delimited by its *interfaces*

3.9
Functional Element
entity of hardware or software, or both, capable of accomplishing a specified purpose of a *device*

3.10
function block
software *Functional Element* consisting of one or more input, output and contained *parameters*

3.11
input data
set of data related to an invocation of a *function block* or a *class*

3.12
instance
entity with unique identity and a set of operations that can be applied to it, and state that stores the effects of the operations [OMG]

3.13
interface
shared boundary between two *Functional Elements* defined by functional characteristics, signal characteristics, or other characteristics as appropriate [IEC Dict 351-11-19]

NOTE The concept includes the specification of the connection of two devices having different functions.

3.14
method
procedure or process for doing something or providing a service

3.15
meta model
model that defines the language for expressing a *model* [OMG]

3.16
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]

3.17
modular, non-modular (device)
physically and/or logically structured (device) as a composed or a single item

3.18
object
1. *instance* of a *class* [OMG]

2. entity with a well-defined boundary and identity that encapsulates state and behaviour [OMG]

   NOTE State is represented by attributes and relationships, behaviour is represented by operations, methods, and state machines.

3. collection of related data and *methods* (procedures) for operating on that data that have clearly defined *interface* and behaviour

3.19
output data
set of data related to an invocation of a *function block* or a *class*

3.20
parameter
1. specification of a variable that can be changed, passed, or returned [OMG]

   NOTE A *parameter* may include a name, type, and direction. Parameters are used for operations, messages, and events.

2. characteristic quantity determining the relationship among variables within a given system [IEC Dict 351-11-03]

   NOTE A *parameter* may be constant or depend on the time or on the value of some system variables.

3.21
parameter attribute
property or characteristic of a *parameter* used to present a *parameter's* value for use

3.22
profile (of a device)
set of data often in graphic or template form used for portraying and applying the behaviour, use and *application* of a *device* in a *class*

3.23
service
specific work performed by a *device* or *object*

3.3
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 Executing System
OMG –Object Management Group
S/W – Software
UML – Unified Modelling Languages
URL – Universial Resource Locator
XML – Extensible Markup Language

4 Overview

The device profile guideline presents at first a short introduction into the all over scope of profiles, precise the subset which is this guideline focus on and introduce 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.

There are three different approaches used in the automation industry, the parameter list model, the function block model and the object orientated model. A special clause provides for the interested readers the model based background of the profile development steps and the template.

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

5 Automation Model and Device Profiles

According to ISO 15745 “Open system application framework 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. All together shall be design and implemented that they are able to co-operated each other. The ISO 15745-1 model contains each component of this system (see Figure 1). The automation devices are a subset of this over all system which is in the scope of this profile guideline. Therefore this profile guideline deals with the device model and device profile part of the ISO 15745-1 only.

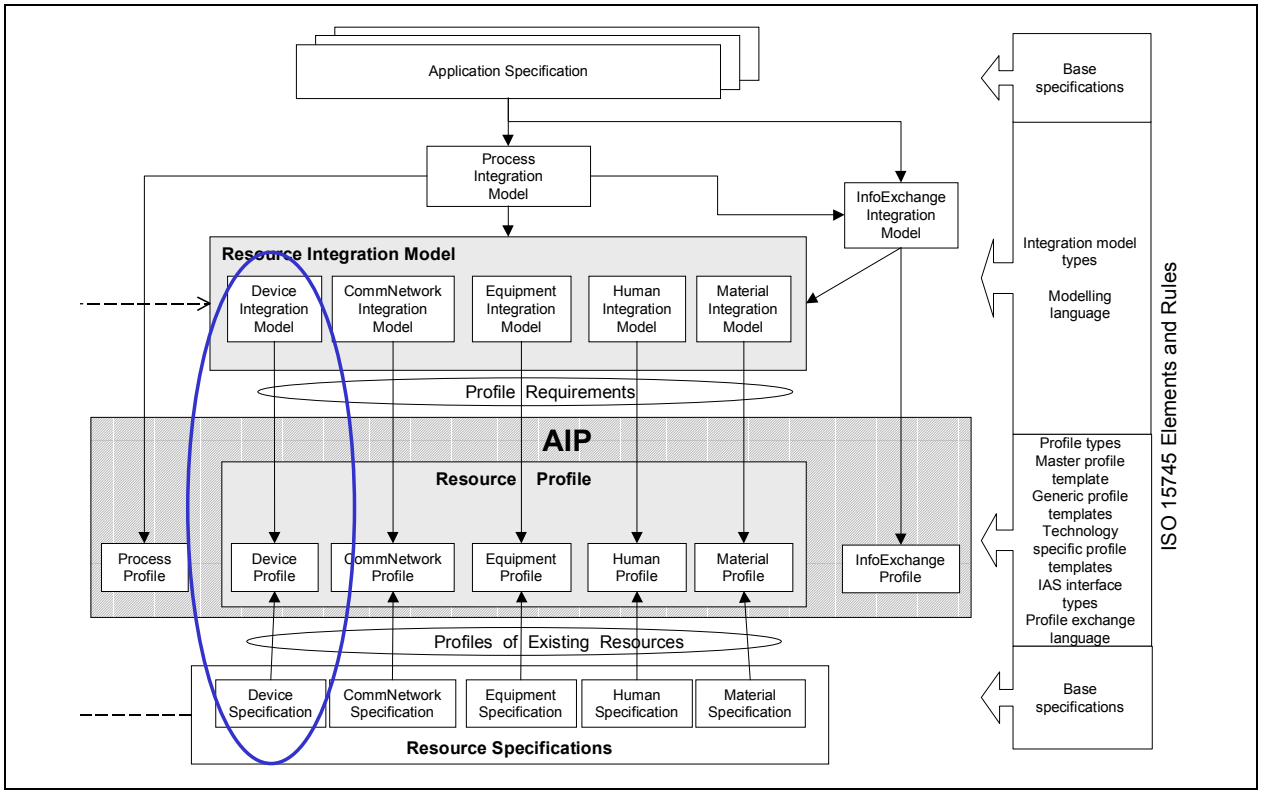

Figure 1 – Profile development using ISO 15745-1

The field device is typically integrated as a component in an industrial automation system. The automation system performs the automation related part of the all over application. To define the complete profile of a specific field device class it is necessary for the profile writers to agree on the required interfaces of the device to the other components of the automation system. The components of an industrial automation system may be arranged in multiple hierarchical levels connected by communication systems as illustrated in Figure 2.

The field devices are the basic components in the process level directly connected via the inputs and outputs to the process (Process I/O).

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

In larger automation systems another higher level may be exist connected via a communication system like LAN or Ethernet. In this 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 Systems (MES), Enterprise Resource Planning (ERP) and other Information Technology (IT) systems can have access indirectly to field devices via the LAN and the controllers or direct via routers.

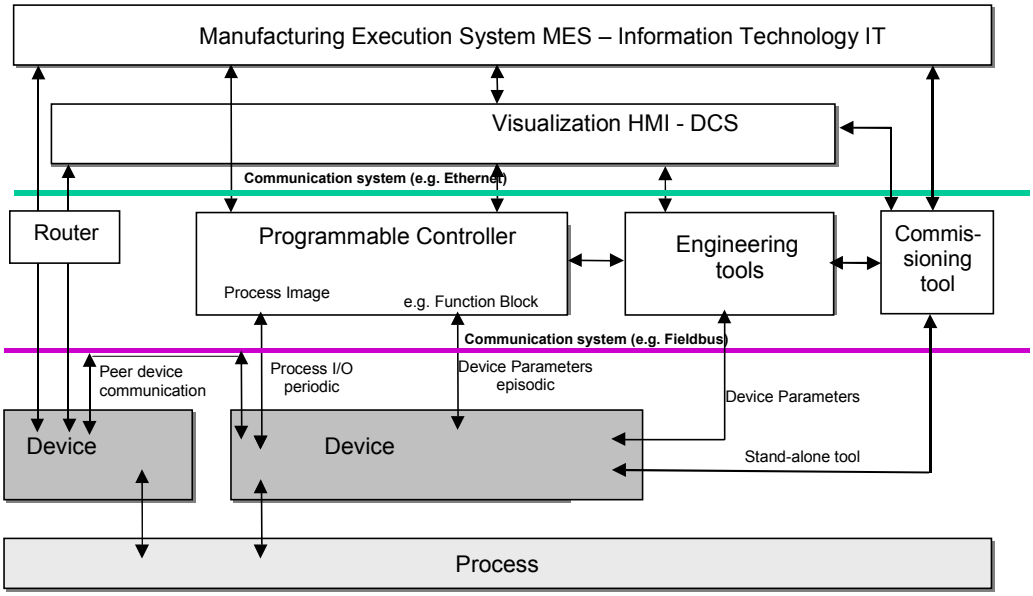

Figure 2 – Typical automation configuration

[Editors note: There has to be a relation between the automation configuration and the terms and models in this guideline; to be done]

The main purpose of devices is to provide certain functions in the all over automation related application. These functions can be structured in a hierarchical way in the device **Figure 3** called “Functional Element” in the scope of this guideline. If the device is modular the functional elements can be both in the modules as well in the device. There are parameters in the device and modules which are related to the functional elements.

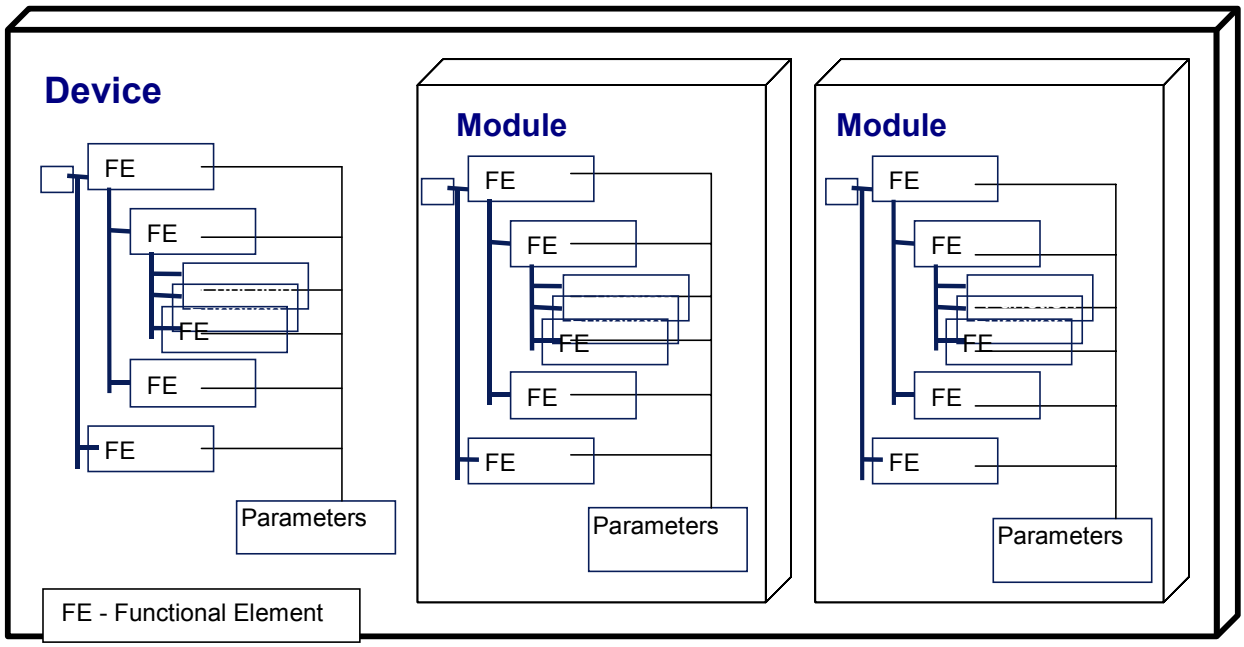

Figure 3 - Modular view of a device's hardware and software

6 Profile definition Steps

6.1 Overview

The developing of field device profiles is a 6 steps process as illustrated in Figure 4. During this process all relevant information has to be collected in the filled-in (completed) profile template. Additionally a profile document with additionally information can be provided. These steps are described in detail in the clauses 6.2 to 6.6.

Each device profile development starts with 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 during the specification work to find decisions. Definitions, normative references and abbreviations are defined during the all over profile development process.

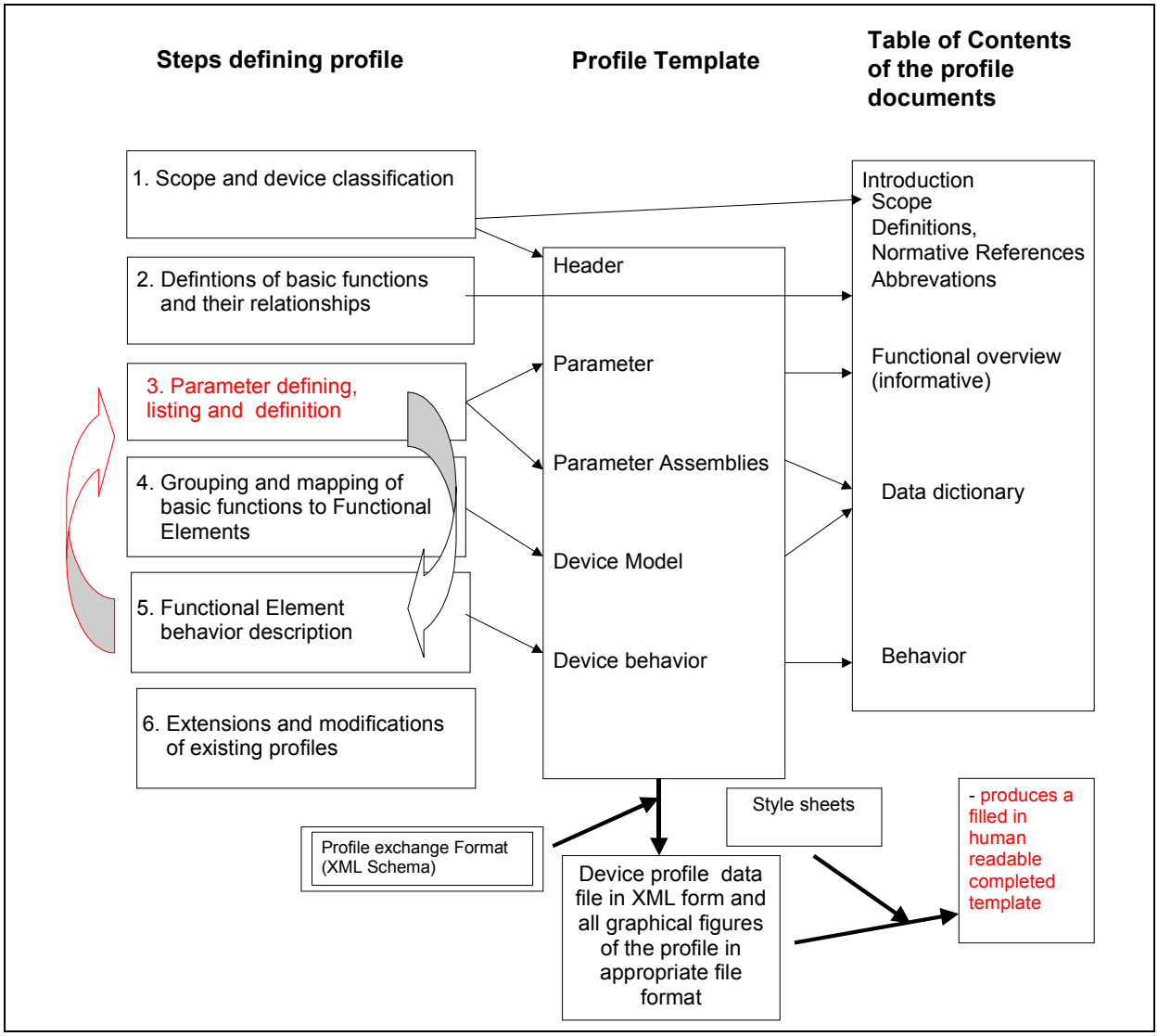

Figure 4 - Profile definition steps

It is necessary to determine 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.

In the third step the data dictionary is the core of the device profile and contains all parameters of the device that are accessible via the communication system. The data dictionary defines the application view to the data but not the communication view.
The fourth step deals with the structuring of the general device functionality into specific functional elements. The functional elements encapsulate the parameters and the behaviour of a device. This step transfers the functional overview developed in the second step into the virtual “device model” which is defined in details in clause 8.
The fifth step is recommended to handle the consistency relations between the parameters and/or the behaviour within the functional elements. For each functional element this has to be described separately in a 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 and description of the functionality of the device.

The sixth step may be applied for the extension and modification 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.

The profile template provides the form in which the profile definitions are recorded. This profile template contains the hard facts, i.e. the main results of the profile development process. There are several users of the profiles. Human beings have to understand the profile. Therefore a profile document is necessary. This profile document extends the contents of the template with explanations and additional text and figures. Device manufacturer and tool developer have to implement the profile definitions. For an unambiguously specification and for a tool based implementation support it would be helpful if there is a machine-readable representation of the profile additionally.

XML provides among others an appropriate technology (Figure 4). The profile template structure is represented in terms of the XML schema. A concrete device profile, i.e. the filled in profile template is represented by an XML file. Style sheets are used to generate out of the device profile XML files different formats which can be used by browsers, text processors or other tools. The details how to use this technology is up to the profile writers or their organisations.

6.2 First step: Scope, compatibility levels and device classification

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) in an automation system. A typical automation configuration is given in Figure 2.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 are collected and defined in the first step.

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 system which are also based on communication profiles provided by communication system organisations (AX, AY, AZ). From the system point of view there are pro-ducts based on different device profiles using one certain communication system (AX, BX, CX). Connecting devices to a system there are N:M combinations of device profiles and communication systems which are increased by manufacturer additions also.

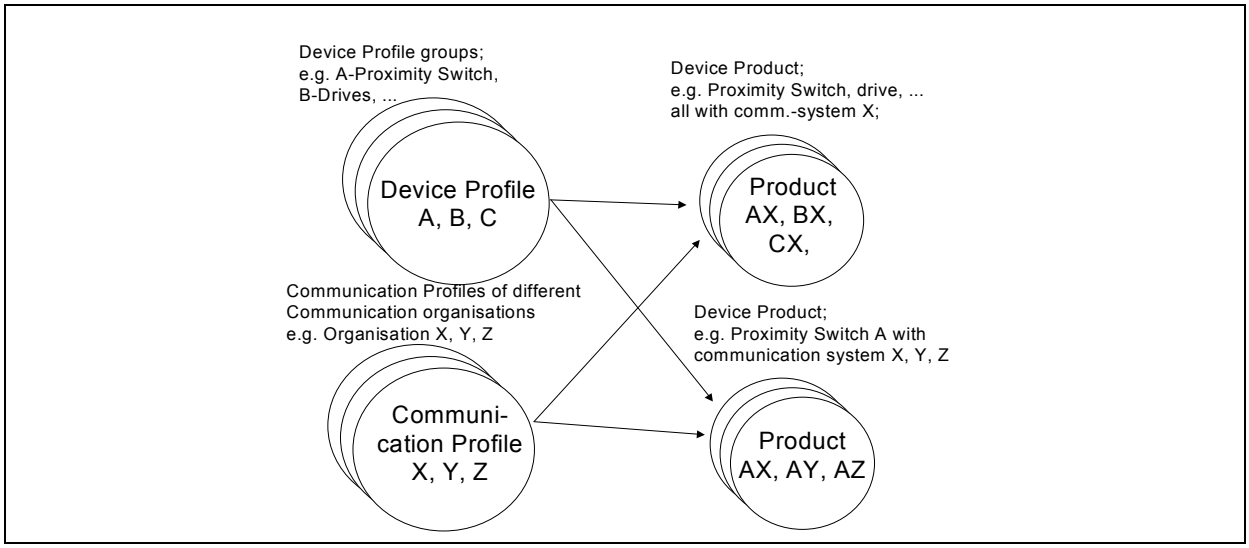

Figure 5 – Relations between profiles and products

There are certain degrees of compatibility and according degrees of co-operation between profiles based devices (compatibility levels). The degree is depended on well-defined communication and application device features.

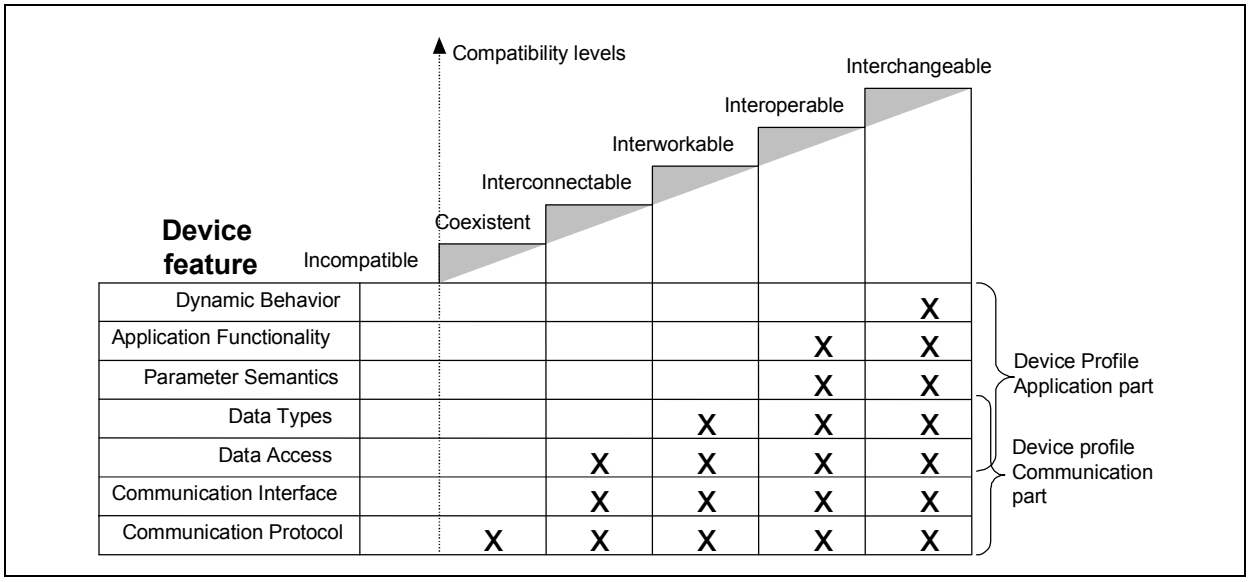

Figure 6 - Levels of functional compatibility

Regarding these device application and communication features the following compatibility levels shall be used for the classification of devices:

Incompatibility

Incompatibility is the inability of two or more devices to work 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.

Coexistence

Coexistence is the ability of two or more devices, regardless of manufacturer, to operate independently of one another at the same communications network, or to operate together using some or all of the same communications protocols, without interfering with the functioning of other devices on the network.

NOTE There have not to be an agreement regarding the communication services.

Interconnectability

Interconnectability is the ability of two or more devices, regardless of manufacturer, to operate with one another using the same communication protocols, communication interface.

NOTE The devices allow data exchange without agreements about the data types. A data type conversion may be necessary.

Interworkability

Interworkability is the ability of two or more devices, regardless of manufacturer, to support transfer of device parameters between devices having the same communication interface and data types of the application data.

NOTE If a device is replaced with a similar one of a different manufacture, it can be necessary to reprogram the application.

Interoperability

Interoperability is the ability of two or more devices, regardless of manufacturer, to work together in one or more distributed applications. The application data, their semantic and application related functionality of each device is so defined that, should any device be replaced with a similar one of different manufacture, all distributed applications involving the replaced device will continue to operate as before

the replacement, but with possible different dynamic responses. Interoperability is achieved when both a field device and a system support the same combination of mandatory and optional parts of the same standard.

NOTE Manufacturer-specific extensions in field devices or systems from different manufacturers may prevent interoperability.

Interchangeability

Interchangeability is the ability of two or more devices, regardless of manufacturer, to work together in one or more distributed applications using the same communications protocol and interface, with the data and functionality of each device so defined that, if any device is replaced, any distributed applications involving the replaced device will continue to operate as before the replacement, including identical dynamic responses of the distributed applications.

There are already separated 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 a taxonomy of measurement and actuation devices. The basic example of automation device classification is provided in annex 1.

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 4 all relevant information of this step are recorded in the header section of the profile template as shown in clause 7.2.8.

6.3 Second step: Definition of basic functions and their relations

Based on a black box model the device is described in a top down approach, 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. SP, measurement value). This first model is detailed by stepwise refining of the main signal flow between device functions. The degrees of details are depending on the device class. It can happen, that different device sub-classes will be introduced at a very detailed level. Devices can have certain subclasses which provide functions for different purposes. These subclasses should be visible.

The overview of basic device functions provides the functional structure of the chosen device class. It is possible that more then one functional diagrams are necessary to cover the device subclasses of the profile.

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

Description method

Functional diagrams are the main description 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 8) for which the device profiles are defined. Use cases lead the profile writer to fill in the content of the profile e.g. parameters and its related behaviour. For more details see A.1

Example Power Drive System functional diagram

Some chosen devices e.g. PDSs (Power Drive System) have hundreds of functions which come into consideration. It is not possible to consider all of them. Therefore, a significant choice has to be done.

Special types of PDSs are for use with an a.c. induction motor. For low voltage applications, the output part of the converter is usually a voltage-source inverter, the most frequently made of transistors (IGBT) and diodes. Such PDS may or may not require feedback signal derived from a sensor mounted on the motor shaft. The control is generally based on a model of the motor (vector control, sliding mode etc.).

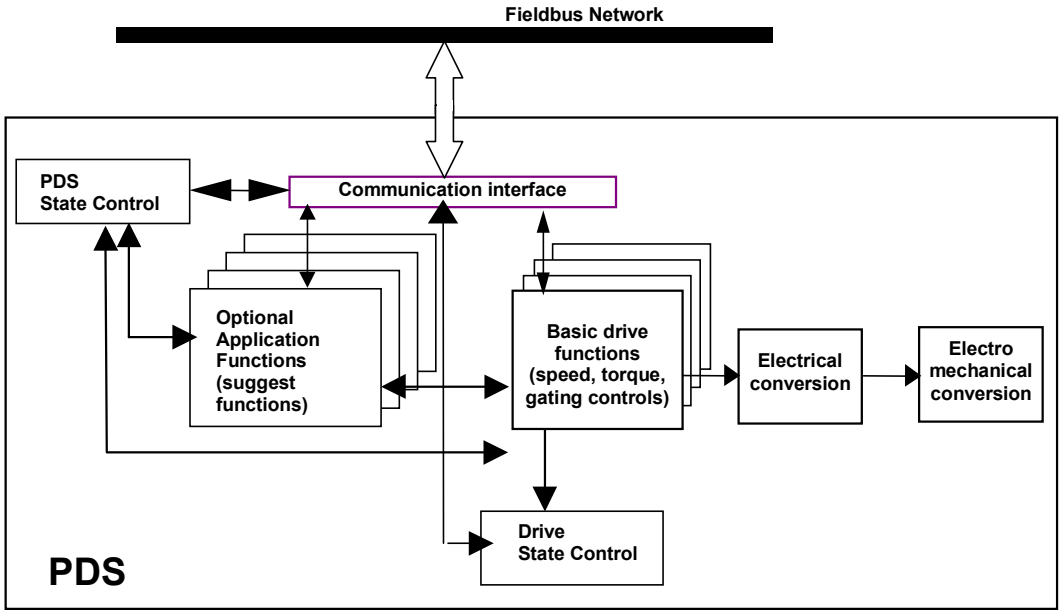

Figure 7 - Functional overview of a Power drive system (PDS)

6.4 Third step: Data Dictionary definition

The data dictionary contains all accessible parameters of a device. The definition of the parameters can follow various procedures, for example:

- Investigations of the device use cases - see Figure 8 and Annex A.1
- Considerations of life cycle aspects - see clause 8.1.3
- Discussion of the device interface in various automation configurations - see Figure 2.
- Derivation of the parameters out of the device functionality – see clause 6.3

These parameters have characteristics (e.g. name and data type) that have to be described as so called the parameter attributes. A representative collection of possible parameter attributes are provided in A.8. One major parameter attribute is the “supported” parameter attribute. This parameter attribute shall be defined always (mandatory parameter attribute). Using this parameter attribute it is advantageous to build profile families (see clause 6.7). Depending on the functional compatibility level the profile intends to achieve (see 6.2), a certain subset of parameter attributes has to be defined.

The data dictionary is recorded in the parameter set section of the profile template.

In terms of using use case investigations for the profile definition the data dictionary and finally the parameter set and the behaviour (see 6.6) of the profile template can be derived out of the found use cases.

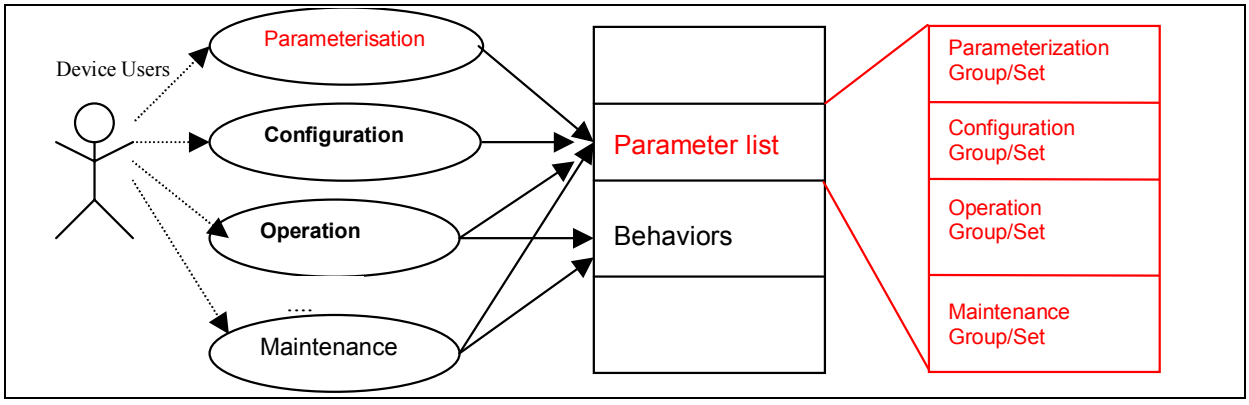

Figure 8 - Use case examples

Use case and scenario definition should follow the definition of UML.

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

6.5 Fourth step: Grouping of basic 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 basic 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 to model and how to handle the basic device functions. These models are described in details in clause 8.3. The profile writers have to decide which model they will use for their profile development:

a. Parameter List device model – see clause 8.3.1
b. Function Block device model – see clause 8.3.2
c. Object oriented device model - see clause 8.3.3.

To offer a common view of these three models the term *Functional Element* is introduced in this guideline. A *Functional Element* is either a single parameter, single function block, or a single object in the device model. The grouping is the encapsulation of functions and device data to Functional Elements. The device data term will be used for the data input/data output/parameter terms of the Function Block model and the attribute term of the object model.

The basic functions become the methods in terms of object model and algorithms in terms of the block model. The functional elements and their relations are the structure of a device class which is recorded in the device model section of the profile template. Modular and configurable devices provide a list of Functional Element types that can be instantiated in the devices. The defined types Device Function Functional Element, Application Process Functional Element, Device Manager Functional Element and Identify Functional Element have to be used (see 8.1.4).

The details of the device model section of the profile template are defined in this profile development step.

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

6.5.2 Example of a Flow transmitter using Function Block and Object model methods

The following two figures, Figure 9 and Figure 10, show the structure of a flow transmitter, which is modelled with function blocks and objects. The dotted ellipse of Figure 9 represents the application objects class with an internal structure. Each sub-object class is specified in detail in the according profile. The Flow Transducer represents the hardware only and is out of the scope of the according profile. Parameter and Assembly object classes are the interfaces

of the application object class to the communication system. Message router and connection object class are communication system specific classes.

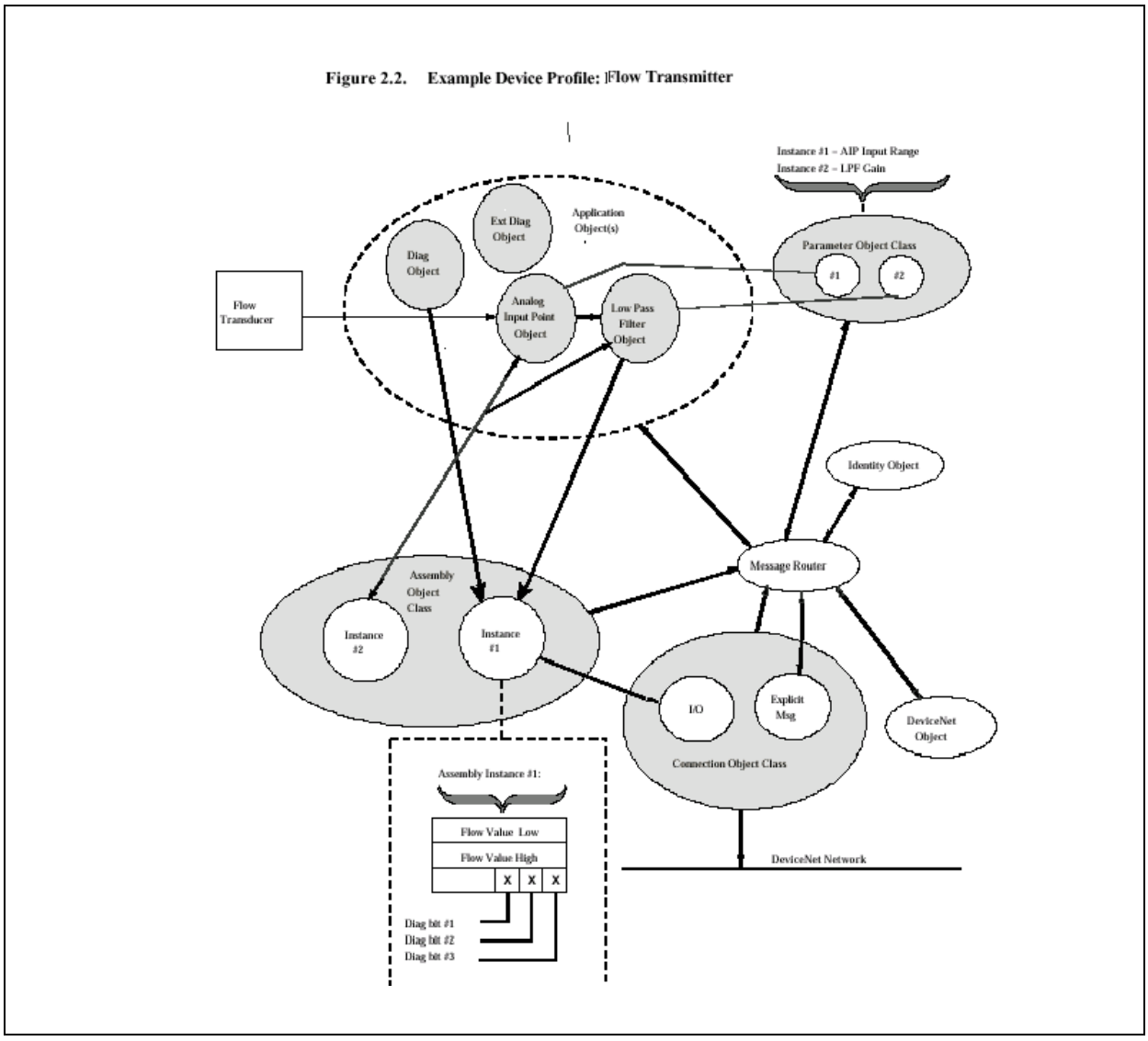

Figure 9 - Object Model of a Flow Transmitter (Example)

The flow transmitter example based on Function Blocks shows a more detailed structure regarding the internal signal flow. The software part of the transducer of the transmitter is modelled as Flow Transducer Function Block. The example profile models three process values i.e. mass flow density and temperature, each with a separate analogue application signal processing. Additionally a Totalizer Function Block count the mass flow in general. Access from the communication services to the function block parameters can be provided directly or using view function blocks.

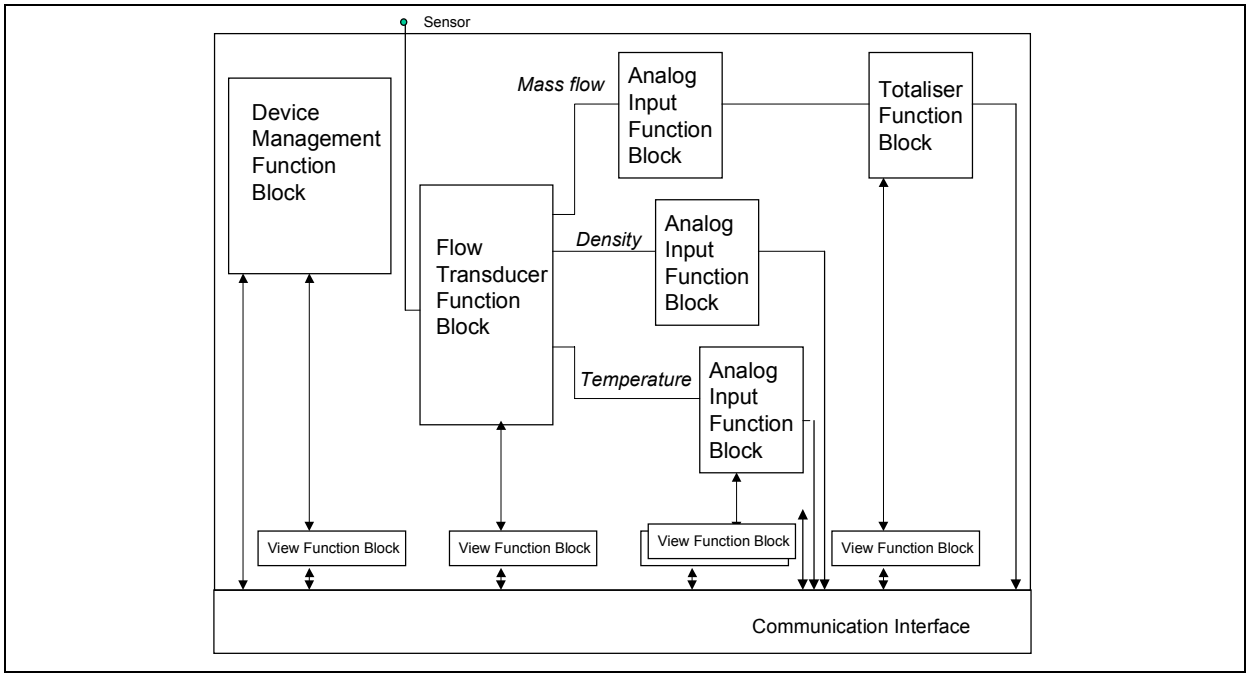

Figure 10 – Function Block model of a flow transmitter (example)

6.6 Fifth step: Functional Element behaviour description

The parameter description of one parameter doesn't cover dependencies between parameters and to the device in general. This has to be considered additionally in terms of the so called behaviour. Examples of such behaviours are state machines, mathematical equations and consistency rules among parameters and conditions (e.g. if ... then ...) that are influenced and reflected in more then one parameter. This behaviour should be described in connection with the Functional Element.

Profiles chose a subset of the over all device behaviour which is necessary for the co-operation of the devices in a distributed system. The behaviour is composed of a set of algorithms and methods respectively. The following algorithm/method types are very common in block and objects respectively:

Algorithms in the mathematical sense e.g. normalize, scaling, filter, invert, which calculate from input data using parameters output data; one possible implementation of these algorithms are IEC 61131-3 functions.

State automata are e.g. operation modus of devices like running, ready, stop and automation functions like Off/On as shown in Figure 12.

Sequences of algorithms including process and communication interactions e.g. alarm handling, calibration, start up phase

Event driven sequences of algorithms e.g. status depended start of synchronization processes or shoot down processes; time depended start of system checks or cleaning processes

The result of this step is the detailed behaviour description referring to the Functional Elements that is the device behaviour section of the profile template.

The third to the fifth step will be carried out iterative because parameter and functions are tightly connected.

6.7 Sixth step (optional): Extensions and modifications of existing profiles

This additional optional step is necessary if a defined set of profiles or devices are derived out of a basic (root) profile. The derived profile can modify the basic profile as followed:

- Add parameters
- Add behaviour
- Add functional elements

Basically all definitions which have been done in step 1 to 5 are considered once again. This is a process leading to a whole set of related device profiles (profile family consisting of several derived profiles).

Note: It is in the responsibility of profile groups or manufacturer to create this kinds of profile families.

7 Profile templates

7.1 General

The developers of device profile standards (profile writer) shall provide a common representation of networked industrial devices. This guideline sets forth a requirement that this representation framework shall be a template for documenting that representation. This template serves as a form that when filled in by a profile writer becomes a human readable device profile. A device profile standard shall provide information on the required device profile template's content. A filled in profile template is the result of profile development procedure described in the profile development steps (clause 6). The template should be exchanges by a profile exchange language such as XML additionally as described in 6.1.

7.2 Profile Template structure

7.2.1 Overview

The profile template shall be organized and divided into sections to aid in the device profiles readability and understanding. The following sections are recommended:

a. a header section, which is filled with the result of the first profile development step, scope and device classifications,
b. a parameter section, which is filled with the result of the forth profile development step, data dictionary definition,
c. a parameter assemblies section, which is filled with the result of the third profile development step,
d. a device model section, which is filled with the result of the second and forth profile development step,
e. a device behaviour section, which is filled with the result of the fifth profile development step, Functional Element behaviour definition.

These profile template sections are detailed in clauses 7.2.3 to 7.2.7. It is the responsibility of the profile writer group to define its appropriate actual representation format of the profile template. An example of such a representation is provided in clause 7.2.8, showing all the profile sections above.

7.2.2 Procedure to fill in the profile template

The procedure to fill in the profile template is illustrated in Figure 11. This procedure is closely connected with the steps described above (clause 6). Step 2 results in a collection of device functions which are under consideration of the device profile. These functions are not recorded in the profile template. Step 3 derives the list of parameters out of the functions while step 4 groups the functions together to functional elements. The parameter are recorded in the parameter set section in the profile template. The parameters can be grouped to assemblies which has an according profile template section. Additionally parameters are associated to functional elements in step 4 which is recorded in the according parameter attribute in the parameter set section of the profile template. An overview about the functional elements of the device is contained in the device model section of the profile template. Simple device using the parameter list model (clause 8.3) don't have to define functional elements. They are seen as devices with one functional element only. Step 3 and 4 as well as step 5 (functional elements behaviour description, which is not shown in the Figure 11) are carried out iterative, i.e. they can be carried out several time and using different orders. The description of the behaviour is a subsection of the device model.

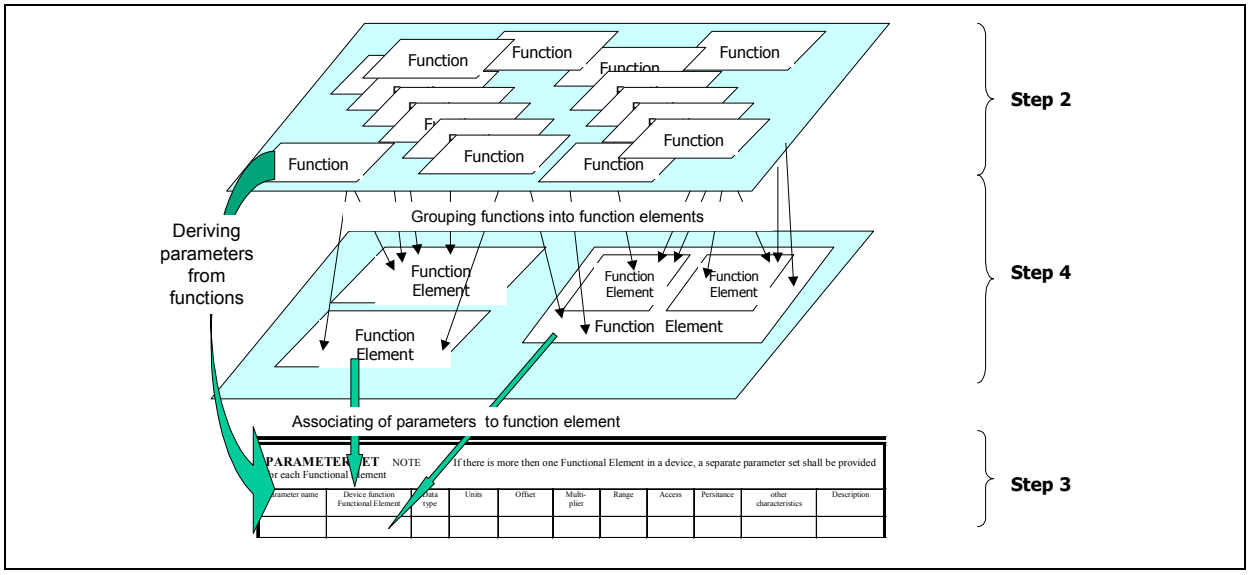

Figure 11 - Procedure for developing device profiles

7.2.3 Header

The header section shall provide device profile identification. It represents the identity functional element. The header shall make it clear to the reader of the profile that he has the correct device profile.

If a profile is for a class of devices (e.g. a profile defined by an IEC product committee, called “root profile” in the example of clause 7.2.8), the contents of the header shall provide an unambiguous identification of this profile, including the identifier assigned to this profile by the profile writer organization, profile revision and profile release date. A field for additional device description information shall be provided.

If a profile is for a specific device (e.g. a profile defined by a manufacturer, called “manufacturer’s profile” in the example of clause 7.2.8), the header shall provide an unambiguous identification of this specific device. This identification information includes elements such as the device’s name, catalogue number, manufacturer, version. A field for additional device description information shall be provided.

The header definitions are a result of the first profile development step, scope and device classifications.

7.2.4 Parameter Set

The parameter section of the template shall provide a list of all the device’s parameters that are accessible in the device through the network. Each parameter shall contain parameter attributes that allow the user of the device to properly provide, use and display a parameter value. The parameter should use the parameter attributes as defined in A.8 where possible to provide uniformity in the parameter description.

Complex devices may have large numbers of parameters. It is recommended that profile writers provide a parameter attribute that identifies the device’s function to which it belongs. The device functions are represented by the Functional Elements like device functions, application process and device manager. A representative example of parameter attributes is included in the example template of sub-clause 7.2.8.

The parameter set definitions are a result of the forth profile development step, data dictionary definition.

7.2.5 Parameter assemblies

The parameter assembly section shall provide a list of groups of device parameters that are available from and accessible in a single network message. Not all networks support parameter values assembled in one network message. The only parameter access is via a single network message per parameter value. Some networks and their devices only support access to device parameter values through parameter assemblies. Therefore the inclusion of parameter assemblies section is not mandatory.

The template shall allow the defining of the parameter mapping in the parameter assemblies.

The parameter set definitions belong to the third profile development step.

7.2.6 Device Model

The profile shall contain a device model which is represented in terms of a graphical diagram and a table. They contain the Functional Elements and their relations.

- Simple devices don't separate Functional Elements so that the device is one single Functional Element.

- More complex devices are a collection of Functional Elements with connections between them.

The device model definitions are a result of the second and forth profile development step.

7.2.7 Device Behaviour

There are state driven and non state driven device behaviour.

The behaviour of each Functional Elements should be described in terms of a state chart; an example is shown in Figure 12. The profile template in clause 7.2.8 contains a state machine represented as state chart, state table and state transition table to describe the behaviour of a simple device. Both representations are recommended because the graphical one is good for a quick human overview and the table one is good for software tools because of its standardized approach. The descriptions of State Machines and State Chart diagrams are given in ISO 19501-1. There are different possibilities to model the state driven behaviour of the device/Functional Elements. UML State Chart descriptions give the opportunity to model hierarchical and parallel state machine in one chart, i.e. the device have one single state machine. It is also possible to decompose the device in more then one Functional Element where each of it has its own state machine. Both description variants are possible.

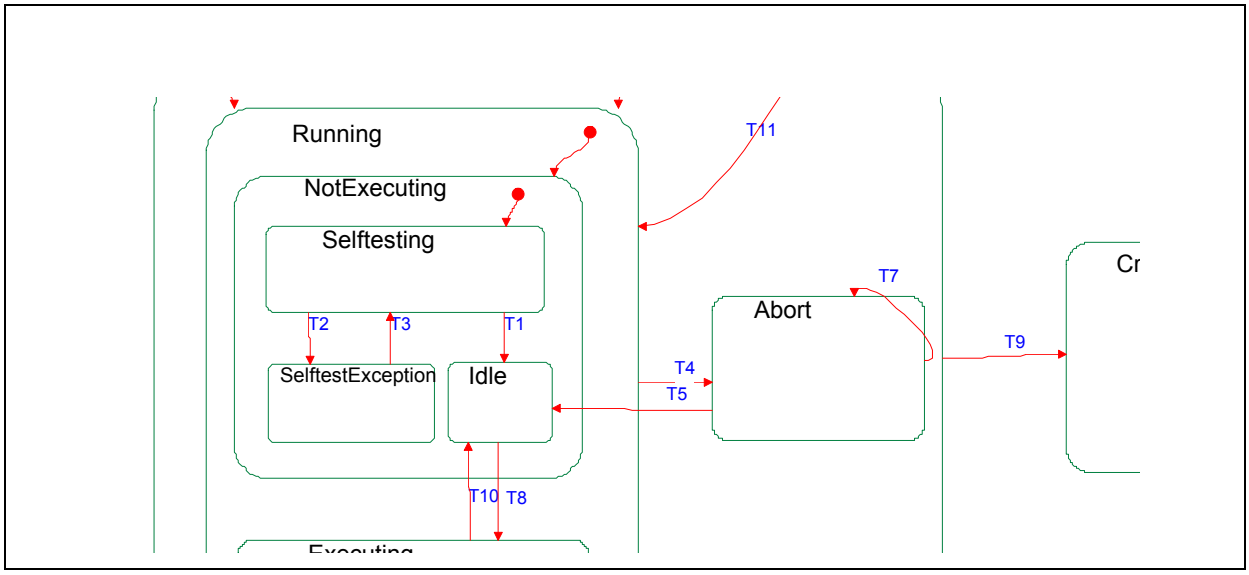

Figure 12 - Example of a device behaviour as state chart

Non state driven behaviour like mathematical and procedural algorithms as well as conditions and constraints should be described in terms of IEC 61131-3 languages.

The device behaviour definitions are a result of the fifth profile development step, Functional Element behaviour definition.

7.2.8 Template form

Table 1 shows an example of a representation format for the profile template.

NOTE This example is derived from the base template specified in IEC 61915, with some modifications/extensions.

Table 1 - Template of a device profile (Example)

MANUFACTURER'S DEVICE PROFILE HEADER

Manufacturer's device profile ID:	Manufacturer's device profile description:	Manufacturer's device profile version:	Manufacturer's device profile release date
Manufacturer ID:	Model compatibility:	Software compatibility:	Hardware compatibility:

ROOT DEVICE PROFILE HEADER

Root device profile ID:	Root device profile version:	Root device profile release date:	
Device profile description:			

PARAMETERS

NOTE If there is more then one Functional Element in a device, a separate parameter set may be provided for each Functional Element

Parameter name	Device function Functional Element	Data type	Units	Offset	Multi-plier	Range	Access	Re-quired	Other pa-rameter attributes	Description

PARAMETER ASSEMBLIES

NOTE If there is more then one Functional Element in a device, a separate parameter assembly may be provided for each Functional Element.

Parameter assembly name:	Access:	Required:	Implemented

Byte	Bits: (0-7 for byte constructions; 0-15 for word constructions)							
	7	6	5	4	3	2	1	0
	15	14	13	12	11	10	9	8
0								
1 …								
N								

DEVICE MODEL

Here: One device function block diagram or object diagram per device.
Simple devices may have one function block or object only.

Functional Element class	Required	Number and purpose of instantiations
Device Manager	Mandatory/Optional/…	1 instance
Device Function		2 instances: Temperature and Pressure
…		

DEVICE BEHAVIOUR

NOTE: If there is more then one Functional Element in a device, a separate behaviour description may be provided for each Functional Element

Behaviour description can be done here provided in different styles such as:

- **Equation**
- **Condition**
- **Constraint**
- **Graphical and textual State chart (see below)**
- **Services**
- **etc…**

STATECHART DIAGRAM

Here:	**One device function block diagram or object diagram per device.** **Simple devices may have one function block or object only**		
STATE TRANSITION TABLE			
STATE NAME		STATE DESCRIPTION	
TRANSITION	SOURCE STATE	TARGET STATE	EVENT

NOTE Example profile in the annex e.g. LVSG and PA Temperature (to be done)

8 Device profile and device model

8.1 The ISO 15745-1 Device Profile definition and its views

8.1.1 Overview

This standard is based on the device profile definition part of the ISO 15745-1. This device profile is specified by the Device Profile class diagram shown in Figure 13a. The specification technique is based on the object-oriented paradigm. A short description of the main elements of object-oriented design and the used Unified Modelling Language (UML) is given in annex A.5. The purpose of the device profile classes is described in clause 8.1.2.

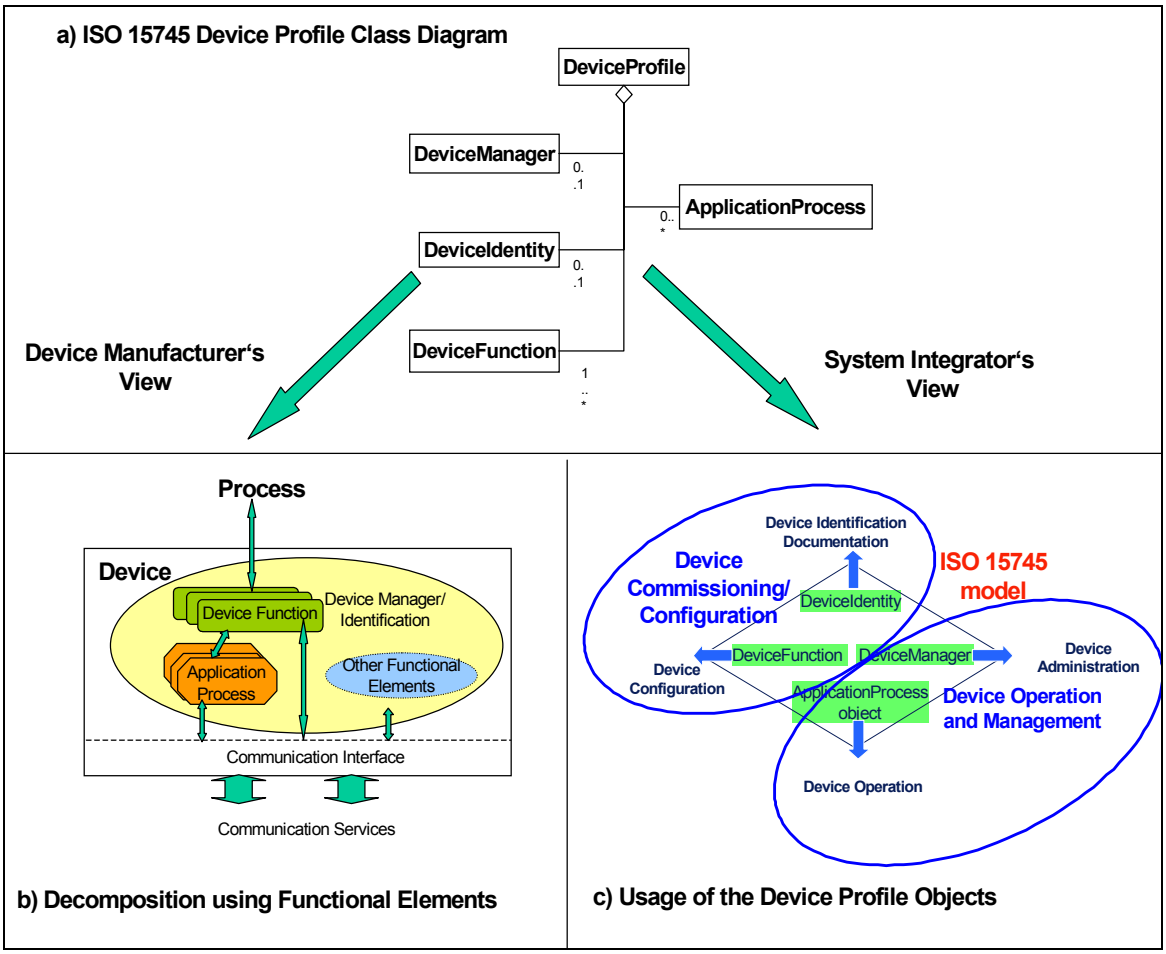

Figure 13 - Views of the ISO 15745-1 Device Profile Class Diagram

8.1.2 ISO 15745-1 Device Profile Class Diagram

8.1.2.1 Device Identity

According to the ISO 15745-1:

A device identity object of the ISO 15745-1 Device Profile Class diagram in Figure 13a contains the attributes of the device that are independent of the network being used and the process being controlled. 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.

Note: Additionally application information like for example installation date, responsible engineer, supported certifications and used capability in this application.

8.1.2.2 Device Manager

According to the ISO 15745-1:

A device manager object of the ISO 15745-1 Device Profile Class diagram in Figure 13a 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.

Note: Typical additional tasks include the Service events (operating hours acquisition) and the Program loader for "Application Objects" in the form of macros, function blocks, etc.

In a distributed application, device administration assumes parts of the all over system administration.

8.1.2.3 Device Function

According to the ISO 15745-1:

A device function object describes of the ISO 15745-1 Device Profile Class diagram in Figure 13a the intrinsic functions 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.

Note: Typical tasks include: Sensor conversion, Sensor value linearization, Counting switching cycles, Calibration, Storage of maximum or minimum values, e.g. maximum internal device temperature, Maintenance, Test functions for predictive diagnosis.

8.1.2.4 Application Process

According to the ISO 15745-1:

An application process object of the ISO 15745-1 Device Profile Class diagram in Figure 13a represents a set of attributes and services that correspond to the application requirements captured in the attributes and services of the associated process profile. Therefore the application process object describes the behaviour of the device in terms of the application, independent of the device technology.

Note: 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.

8.1.3 Views to ISO 15745-1 Device Profile Class Diagram

Several actors have different views to this abstract device profile class diagram during the all over device life cycle in different ways. In Figure 13c the typical two views are illustrated.

The device manufacturer sees this classes as a device decomposition into Functional Elements. Each Functional Element fulfil a determined purpose regarding the device tasks and functionality which are for example related to the operating system of the device, the transducer hardware and software or the application functions which are modified regarding the process condition. The Functional Elements are objects (i.e. instances) of these classes.

The system integrator has during the commissioning phase to parameterize, to configure or to program the device in order to adjust the device to the specific requirements of its purpose in the process. The system integrator does not see the structure of a device divided into Functional Elements. He most of all makes use of the Device Function and Device Identification objects for his tasks. From the system configuration and operation point of view the Device Management and Application Process objects are used.

8.1.4 Decomposition using Functional Elements

The Device manufacturer has a functional driven, hardware/software view to its device. In Figure 13b the abstract device profile class diagram definitions are represented in a way which is more familiar to the manufacturers. Therefore the device profile classes are represented as Functional Elements in this guideline.

8.1.5 Usage of Device Profile Objects

The Figure 13c the system integrator's view shows the typical usage of the main parts of the Device Description / Device Profile from the ISO 15745-1 model and which objects are used by the device tools and which objects are used by the system tools.

The device commissioning tools are directly concerned by the Device Identity object, for example user selection guides or user's documentation. Some device configuration and device tuning are based on the device function object information.

From the runtime perspective, the systems tools are directly concerned by the Device Manager object and the Application Process object to configure and monitor a device integrated into an application system.

8.2 Device parameters and behaviour

8.2.1 Linking to ISO 15745-1 Device Profile Class Diagram

[Editors note: This clause is still under discussion and don't represent the intention of all working group members]

The device profiles according this guideline defines parameters and behaviour which are encapsulations of functional elements. This circumstances is an aggregation relation between the functional elements and the parameters and behaviour, shown in the class diagram in Figure 14.

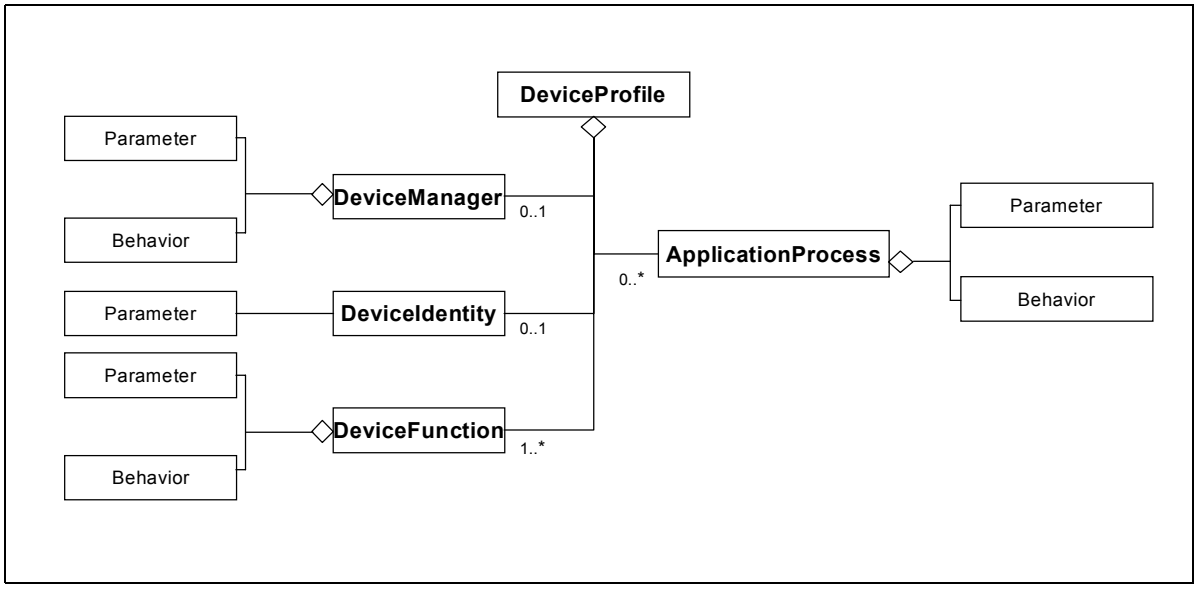

Figure 14 - Linking to Device Profile Classes Diagram of ISO 15745-1

The parameter definitions are based on the Data Dictionary in sub-clause 8.2.2 and the behaviour definition which is based on description in sub-clause 8.2.3.

Note: For the purpose of this guideline the terms parameters and behaviour are linked to the Device Profile Class Diagram of ISO 15745

8.2.2 Data Dictionary

The data dictionary is a conceptual idea. At the first step the data dictionary is a description of the parameter attributes of the Functional Elements of the device. The data dictionary is a schema in which for each data a certain set of parameter attributes has to be defined. The data dictionary is represented in this guideline by certain section of the template. This is a table where each row is one parameter and the columns are the parameter attributes of it. Table 8 in A.8 defines a superset of parameter attributes that can be defined in a profile. Each Functional Element covers a separate part of the over all data dictionary of a device.

The data dictionary is the model supporting the Profile Template which is described in clause 7.

The data dictionary can have different implementations:

The Functional Elements are implemented in a way that their application data are seen via the communication interfaces according to the data dictionary. For this purpose the data dictionary is a sheet of paper only.

The data dictionary or a part of it is represented in a device description file that is purchased together with the device. This file is used by a DCS for engineering tool to interact with the device. The tools are configured with the details of the devices using this file. For this purpose the data dictionary is the basis of the used device description language.

The data dictionary is implemented in the device and can be read out via the communication system. That means that the tool can be configured reading out the device details in a very consistent way, i.e. the revision of the data dictionary and the device are consistent. For this purpose the data dictionary is the basis for the implementation in the device.

The relations of these parameter attributes to the compatibility features are described in clause A.2.

8.2.3 Device behaviour

8.2.3.1 State model

Functional Elements can be seen as objects of the Device Profile classes. Objects are characterized by its attributes and behaviour. In the scope of this profile the behaviour of an object with a state driven behaviour is described by a finite state machine, e.g. as a UML state chart.

State Model

A *state model* aids in understanding the device or block/object behaviour. A state model shall include a state diagram showing the states and the transitions, a state table describing the characteristics and a state transition table describing the details of a transition, i.e. event, conditions and actions. Any state that is visible through the network shall be defined. The UML state chart definitions shall be used. [OMG: UML V1.3, March 2000, section 2.12]

In the state model the following general rules are used for the states and transitions:

States are shown as rounded rectangles.

Each state has a unique name.

States may be nested within other states.

The initial state (or sub-state in a state) is indicated with an arrowhead from a single dot.

Transitions from a state with nested sub-states specify that all sub-states be exited.

There may be multiple independent and concurrent sub-states in a state.

Transitions between states are shown as lines with arrowheads connecting states.

Each transition has a unique number/identifier.

The description have to be done using the profile template as described in clause 7.

8.2.3.2 Mathematical and procedural behaviour

Functional elements can have mathematical equations, procedural and consistency rules among application data and conditions that have to be described. This behaviour shall be described in a well understandable manner. IEC 61131 should be the first choice.

8.3 Mapping of ISO Device Profile Class Diagram

8.3.1 Overview

The above described device model is seen from an abstract point of view. In principle there are three mappings to it as illustrated in Figure 15.

In Figure 15b the most simple mapping of the device model is represented. Simple devices don't offer the full range of the described functionality necessarily for a public access, i.e. only a small subset of the functionality of the functional elements are visible. Therefore these devices are providing some functions only to other components of the system which are normally not grouped. Profiles of these kinds of simple devices are therefore usually lists of parameters only. Naturally the parameters represents device functions such as scaling and sensor selection, but they give only the opportunity to select out of a certain list of function variants or provide necessary functional parameters. Profile development usually starts with this kind of model which is furthermore advanced to one of the following function block or object model.

The Figure 15c shows the *Functional Elements* mapped to *function blocks*. The function block oriented view comes from the automation approach that is visible in IEC 61131-3 and DCS. The behaviour of the blocks is represented by inputs, outputs and parameters that influence or are influenced by the block algorithms. The modification of inputs and parameters controls the block behaviour.

The Figure 15d shows the *Functional Elements* mapped to *objects* of the information technology. Automation systems and its tools increasingly use information and communication technology. Therefore, the device will also be seen as a collection of objects coming from the object-oriented analysis and design paradigm. The object behaviour and the related attributes define the class of objects. Activating the operations of the objects influences the object behaviour.

The following reasons lead to the use of the object technologies in automation:

Field device functions are/become members of the all over DCS and its engineering. Therefore they have to be managed as DCS objects as well.

Objects are based on the well-defined object-oriented model.

For future systems it is desired that DCS and field devices be based at the same paradigm. The object model seems to be one of the favourite approaches.

The vertical information flow from field device to DCS, MES end ERP will become simpler (DCS – Distributed Control System, MES – Manufacturing Execution System, ERP – Enterprise Resource Planning).

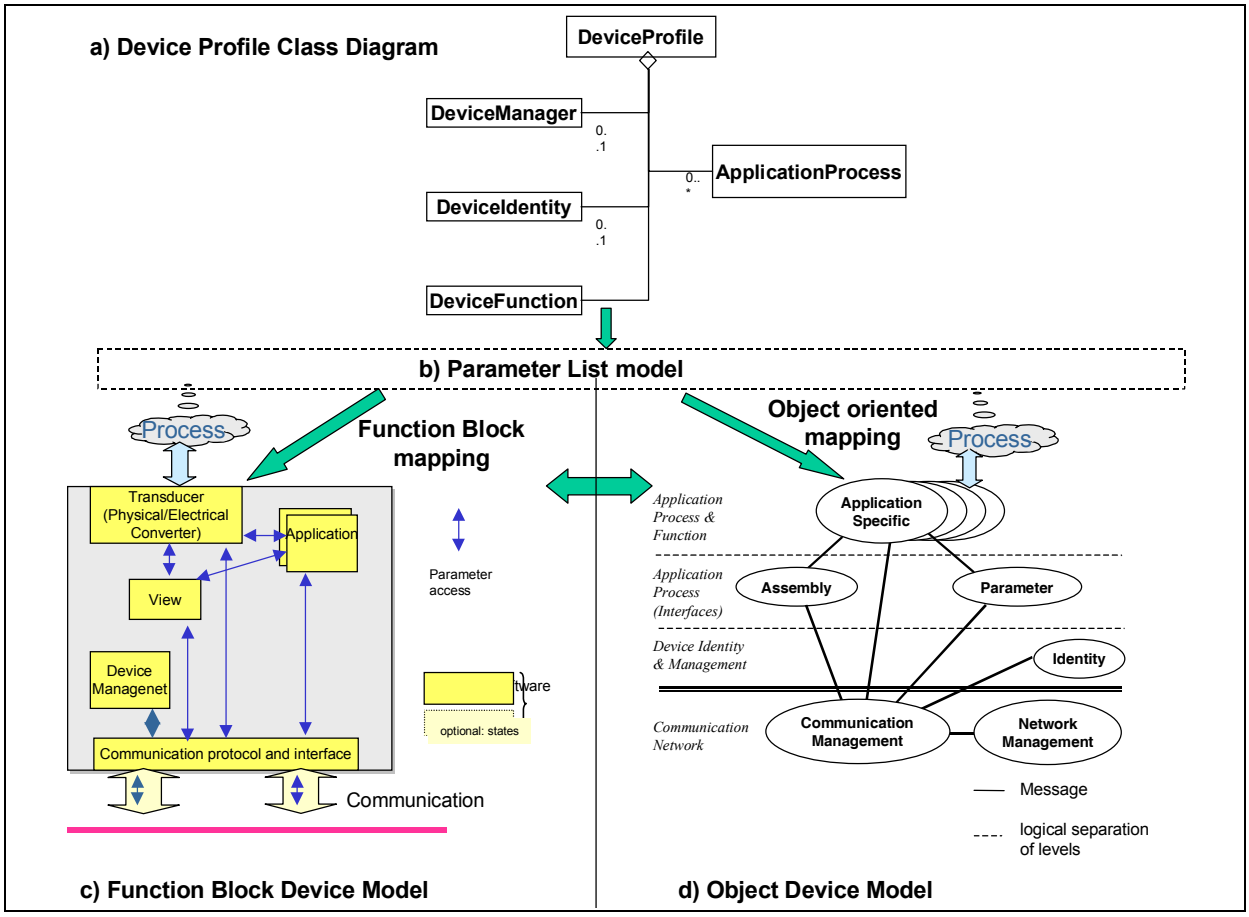

Figure 15 – Function block and object oriented mapping of the device profile

Both models, the function block and the object oriented model follow the object oriented concepts in terms of:

Encapsulation of data and functions

Instantiations of classes to objects; in the function blocks model this aspect is called type and instance

Class/object hierarchies

Inheritance (partly)

Polymorphism

8.3.2 Parameter List Model

The parameter list model consists of parameters only. These parameters are put together as a list where each list entry represents one parameter. The all over parameter list is the data dictionary of the device. The profile template may consists only of a header, parameter list and optional assembly. A graphic device model is not required.

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

8.3.3 Function Block Device Model

8.3.3.1 Background

The basic concepts introduced above are normally not visible for a user of devices that are based on the device profiles according this standard. The profile writers shall only apply these concepts. But the typical profile writer is not used to apply the theoretical device model as introduced above. He may better understand the well-known models as the Programmable Controller programming in IEC 61131-3 languages. Therefore, the two different introduced mappings may be necessary.

Function Blocks are encapsulations of algorithms processing input data and parameters and calculating output data. The data inputs, data outputs, parameters and algorithms are required for designing the process and its control system. They can be derived from the *Pipe and Instrumentation Diagram (P&ID)* that is illustrated in Figure 16. The Function Blocks perform the control and monitoring application. *Function Blocks* are used to structure the control application in a modular and hierarchical way. They are widely used both in manufacturing and process control.

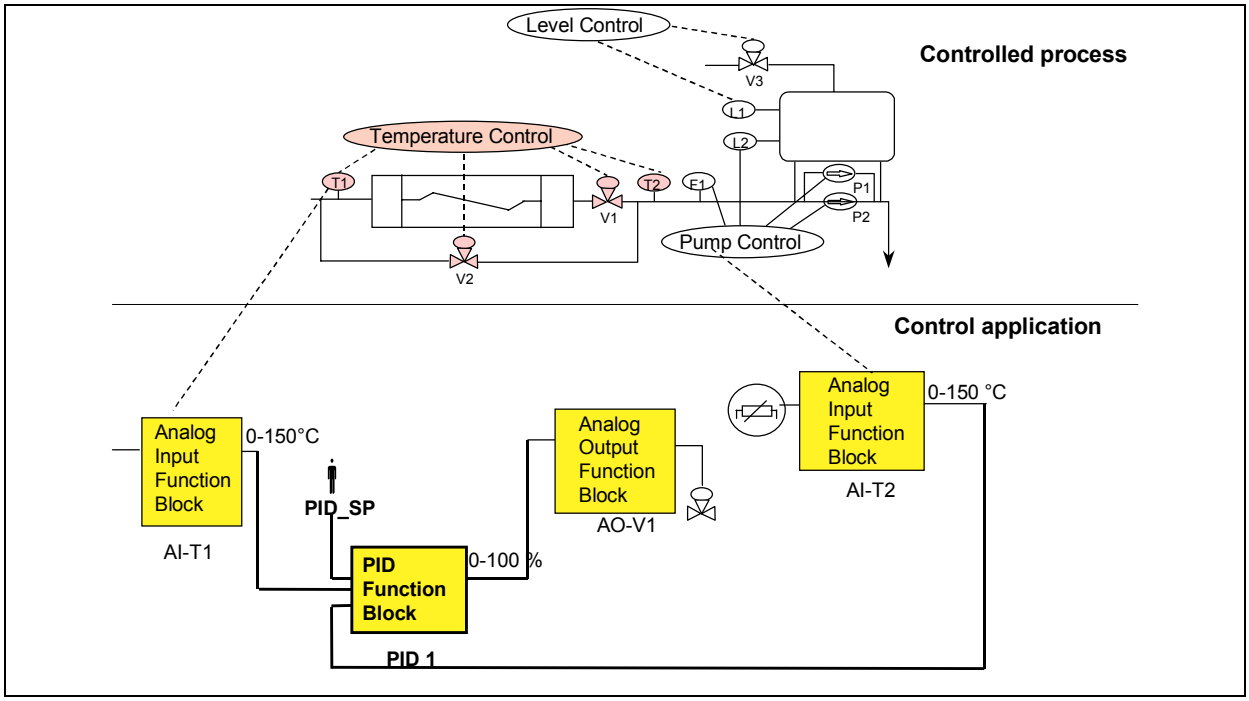

Figure 16 - Function Block structure is derived out of the process

Function Blocks are abstract representations and may be implemented in different ways in different device types according Figure 17.

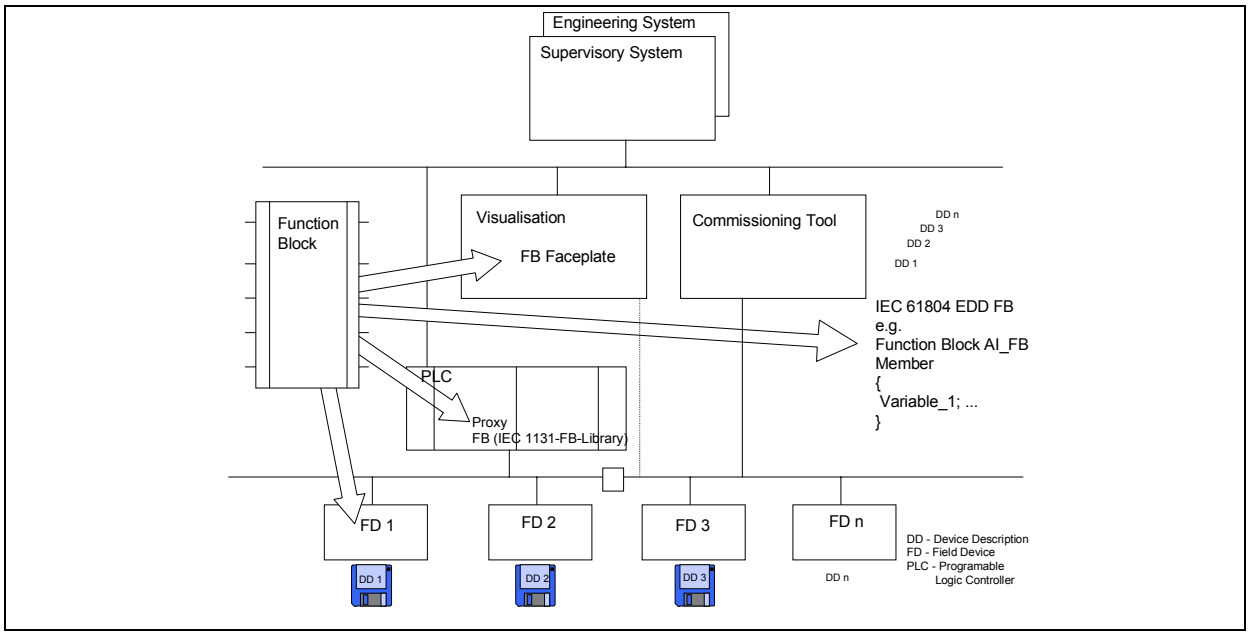

Figure 17 - Function Blocks can be implemented in different devices

Programmable Controllers are using *Function Blocks* in the IEC 61131-3 language Function Block Diagram (FBD) to program the control application. Basis function block types of the IEC

61131-3 are for example RS Flip Flop, Time Delay and others. This where in former times implemented in hardware, i.e. in different devices.

Function block networks encapsulated in more complex function blocks and manufacturer specific function blocks programmed in any language built libraries of Programmable Controllers. Control applications can be built out of these modular and hierarchical libraries. These libraries can be structured in an object oriented way.

Digital signal and information processing have increased the functionality of field devices. Additionally digital industrial communication systems extend the single signal connection between field devices and Programmable Controller and among field devices to a broad interface. From the design point of view the differences between control application and field device applications are disappearing. Therefore it is of favour if the field device functionality is seen as function block or the field device application is structured also in terms of function blocks. This is the main paradigm of IEC 61804 function blocks.

Digital processing and communication system together perform a distributed system as illustrated in Figure 18. This is the reason why structuring a control application in terms of function blocks is not sufficient. A system wide execution control shall to be available. Therefore, IEC 61499 specifies an event control flow additionally to the control flow. This event flow controls the time and order of the function block algorithms execution. Then there is no difference from the control application point of view between Programmable Controller and field device any more.

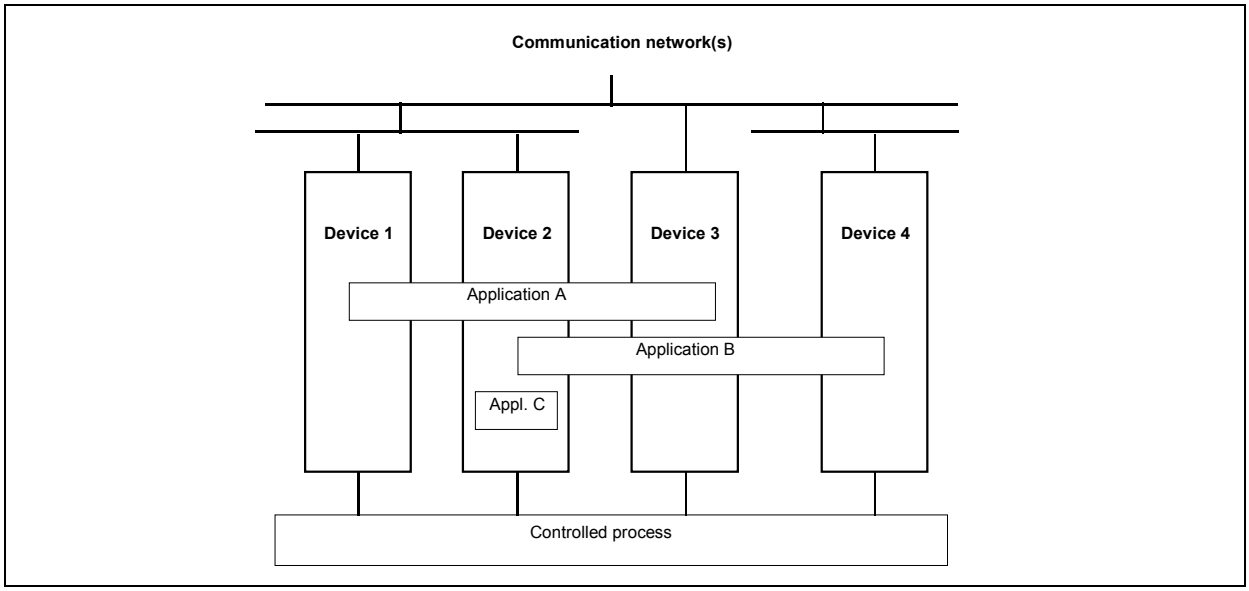

Figure 18 - Function Block structure distributed between devices according to IEC 61499

There is an intermediate step between centralized and fully distributed control applications. A “proxy function block” in the controller represents parts of the control application that are carried out in field devices and not in the controller. This gives the Programmable Controller programmer the opportunity to develop the program in a centralized way. The proxy function block and the field device functionality, represented in terms of input/output data, parameters, function blocks or object, interact in a profile specific or manufacturer specific way.

The visualization face plate is a mirror of a function or function block. The input/output data and the parameters are attributes of the symbols.

For commissioning reasons function blocks are represented in the device description component of a field device such e.g. those of IEC 61804.

8.3.3.2 Function Block oriented Device Model

Using the function blocks paradigm and the device decomposition of clause 4 the function block oriented device model is shown in Figure 19.

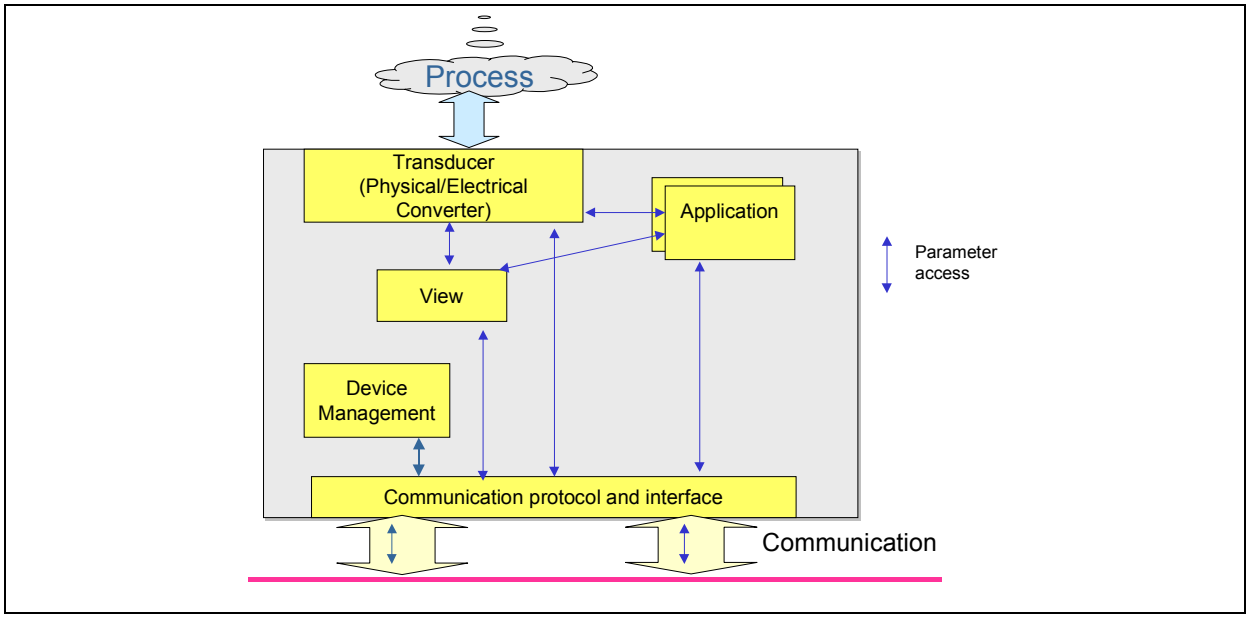

Figure 19 – Function Block device model

The Functional Elements defined in subclause 8.1.3 become device management function block (from Device Management and Identification), electro-physical transducer function block (from Device Function) and application function block (from Application). Collection of input data, output data and parameters are accessible via the view function blocks. The interfaces of the blocks have a type specific mapping to the communication interface, i.e. each data input, data output and parameter has a determined mapping to a selection of communication services. This is shown as arrows in the figure.

Device Management

This part of the device can contain a necessary operating system or a state machine, e.g. controlling the operation modus of the device in general. Furthermore, the Administration controls the access privileges, places diagnosis requests, and supplies diagnostic data. As a convenience function, the Administration provides the complete device information. The Meta Data provides the opportunity to read out the data dictionary with additional information for user guidance via a directory. This includes the Identification information ("Boiler Plate") about device type, manufacturer, versions, reference (URL) to the device's Internet site, etc.

Typical additional tasks include:

Service events (operating hours acquisition)

Program loader for "Application Objects" in the form of macros, function blocks, etc.

NOTE In a distributed application, device administration overtakes parts of the all over system administration.

Electro-Physical Transducer

The field device Functional Elements with immediate access to the automation process such as sensor, measuring unit, or actuator are accommodated here.

Typical tasks include:

sensor conversion

sensor value linearization

Counting switching cycles

Calibration

Storage of maximum or minimum values (e.g. maximum internal device temperature)

Maintenance

Application

In the course of increasingly powerful processors and falling prices, more and more functions are relocated from central computer units to field devices. In the extreme case, there are complete programmable controllers; in the simplest case an application Functional Element transfer the I/O data between bus and transducer block.

View

There is a special concatenation of function block parameters called view. Using views minimize the communication access frequency to the field device. The way of collection data in views and the access strategy is communication system dependent.

8.3.4 Object oriented device model

8.3.4.1 Background

8.3.4.1.1 General

A device may be modelled as a collection of objects. An object provides an abstract representation of a particular component within a device: object modelling organizes related data and procedures into a single entity.

NOTE - The realization of the abstract object model within a device is implementation dependent. A device internally maps this object model in a fashion specific to its implementation.

8.3.4.1.2 Object elements

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 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 change 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 the object interfaces with the communication system.

8.3.4.2 Object classification

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. Multiple object instances within a particular class can reside in a device.

Object elements can be uniquely identified within a device by using a class identifier, an instance identifier and if needed an attribute identifier.

The figure 8 shows the device object model.

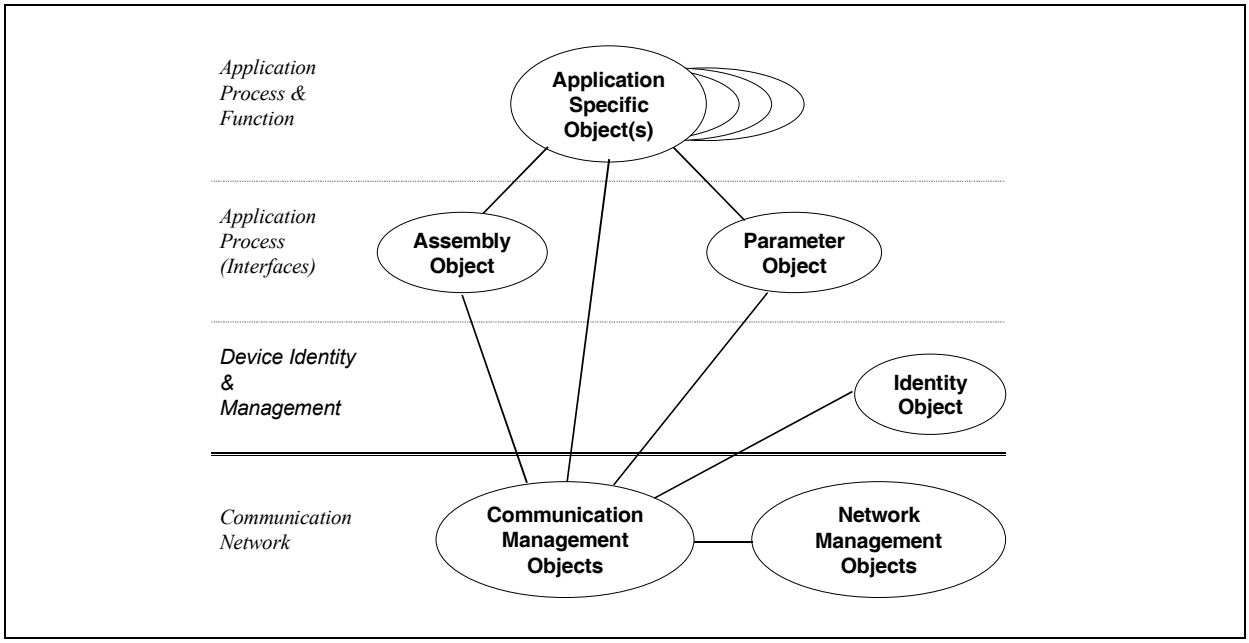

Figure 20 – Device Object Model

Application specific objects 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, e.g. Electro-physical transducer, if there is a need to exchange information related to this device technology, e.g. specific calibration.

Assembly and Parameter objects are used as interfaces between the Application objects and the communication system. Assembly objects allow an optimized access to object attributes by gathering attributes from several objects into a single object: they are typically used for I/O exchanges. Parameter objects are used to provide a public interface to device data: they contain all the information needed to interpret or display an attribute of a given application object.

The Identity object 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.

Communication objects are used as interfaces with the network.

8.4 Comparison of function block and object device model

There is a strong equivalence between the block and object oriented device model which is shown in Table 2.

Table 2 - Reference between block and object oriented device model

Device Profile objects	Parameter List model elements	Block oriented model elements	Object oriented model elements
Device Application Object	-	Application Function Block	Application Object
Device Function Object	-	Transducer Function Block	
Device Identity	-	Device Management Block	Identity object
Device Manager (NOTE)	-		
		Function Block parameters	Parameter object
		View Function Block	Assembly object,

NOTE For Device Manager specialisation see Figure 2

Beside different naming conventions for administration/identity and no separation between transducer and application in the object approach there is a main difference regarding the data access. Objects allow data access by using operations. Therefore there is a preferred mapping of the Application object attributes to assemblies for periodic process related data and to parameters for episodic accesses. Assemblies and parameter objects in the object device model can be seen as interface object with the communication system versus application data in the function block model.

8.5 Modularity (non-modular and modular) for H/W and /or S/W

The defined device model provides a functional modularity. Additionally hardware modularity and a mixture of hardware and functional modularity are available on the device market. Therefore different device types are distinguished based on the aspect of functional flexibility. The basic structure elements of these device types are the Functional Elements and the module.

Module

Module is a *Functional Element* that can contain one or more then one Functional Elements. The device offers slots to plug in these modules. Slots are the places where the modules are plugged in the device rack. In that sense Slots offer the capacity of a device to plug in modules. Additionally modules can built a hierarchy of modules.

NOTE IEC 61131-3 and IEC 61499 define the term resource to provide this kind of modularity. The term resource isn't used because modules can be IEC 61131 or IEC 61499 resources or modules of remote I/O without configuration and programming possibilities.

A compact device has a fixed number and allocation of Functional Elements within modules and modules within slots.

A hardware-modular device has a variable allocation of different types of hardware modules within slots.

A software-modular device has a variable Functional Element allocation to modules. A software-modular device permits modules to be configured although there are no physically plugable units provided.

Mixed hardware/software modular devices combine the features of hardware and software modular devices

Annexes

A.1 General aspects for profile specification

Expert groups of device manufacturers, system providers and users together are specifying device profiles. These groups of device profile writers are the main audience of this guideline. Device profiles are the necessary basis that devices reach a certain level of functional compatibility such as interoperability and interchangeability. System integrators and end users can dramatically reduce the overall engineering costs working with commonly accepted profile-based devices. Prerequisites are high quality profiles which covers all phases of the device life cycle in the system.

The following list of aspects has to be considered by the profile writers. These aspects describe roles of the field device in general use cases of the field device in the all over life cycle:

Devices take place in the commissioning and maintenance role, these roles bring and keep a device out of the stock into operation:

- Configuration

  determines the hardware (e.g. instantiated module) and software (instantiated objects/blocks) structure and their connections of a device
- Parameterization

  determines the initial/default and process related instance values of the modules, objects and blocks
- Working point adjustments / calibration

  determines the relation to the real process (i.e. physical, biological, chemical) values of the measurement and actuation
- Structural information for design and engineering

  Advanced devices provide a directory (table of contents) of the device which can be read out via communication system by tool or other devices
- Commands and Management functions

  Devices have different states and operation modes. These states and modes can be changed by operators or by the devices themselves.

A device is integrated in a system to perform its main aim

- Control application

  The device provides measurement values, get actuation set points and others in co-operation of a control program in a Programmable Controller or DCS.
- Visualization / HMI

  The device provides measurement values, take actuation set points and others in co-operation operator visualization and operator interactions.

A device takes the role of providing information and functions for the service of the device itself:

- Firmware update
- Identification (i.e. device and device configuration identification)
- Diagnosis (i.e. device diagnosis)
- Statistics/trends (of device activities and process values)
- URL (pointer to Home-Page of the device)

Profile writers determine with the details of the profile specification which of the mentioned roles of a device are covered and which aren't covered.

A.2 Level of compatibility as result of a profile specification

[Editors note: This Annex is still under discussion and don't represent the intention of all working group members]

Device Profiles are a set of definitions of application process, device function, device management and identity functionality of automation devices, e.g. field device functions such as process data acquisition, operation mode of the device or limit checking, which communicate with application functions of other devices in a distributes system. The device applications use a specific subset and configuration of the underlying communication system - also called communication profile.

The profile definitions are used for the design, operation and implementation of field devices, for the design and implementation of applications, for instance in a Programmable Controller or DCS and for the design, commissioning and maintenance in the engineering stations as illustrated in Figure 21. Field device, controller, DCS, HMI and engineering stations perform distributed applications in which the device application functions co-operate each other.

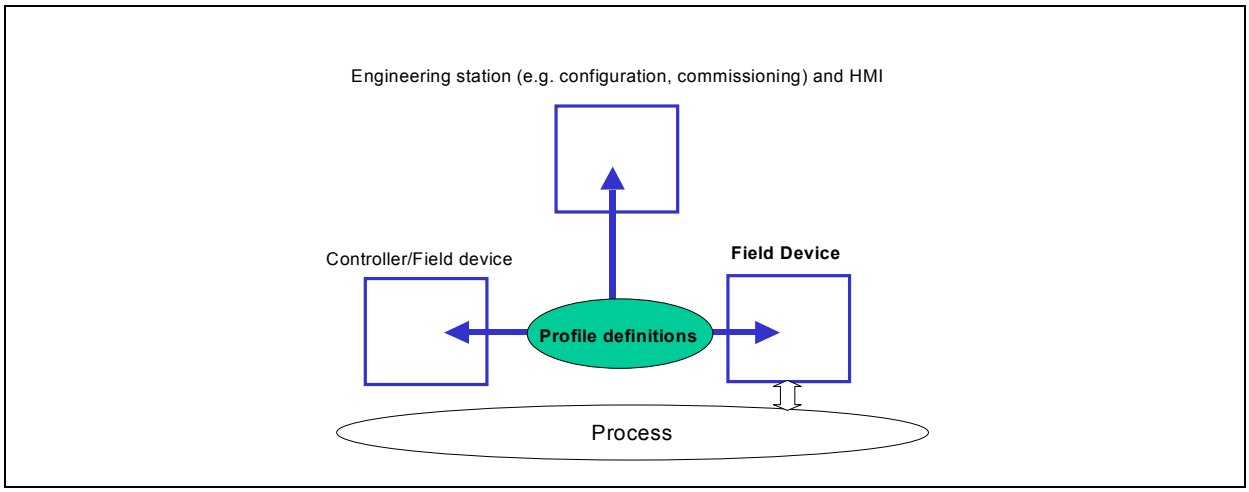

Figure 21 – Usage of Device Profiles in a system

Device profile definitions provide certain degrees of compatibility and according degrees of co-operation between profiles based devices. The degree is depended on well-defined communication and application features.

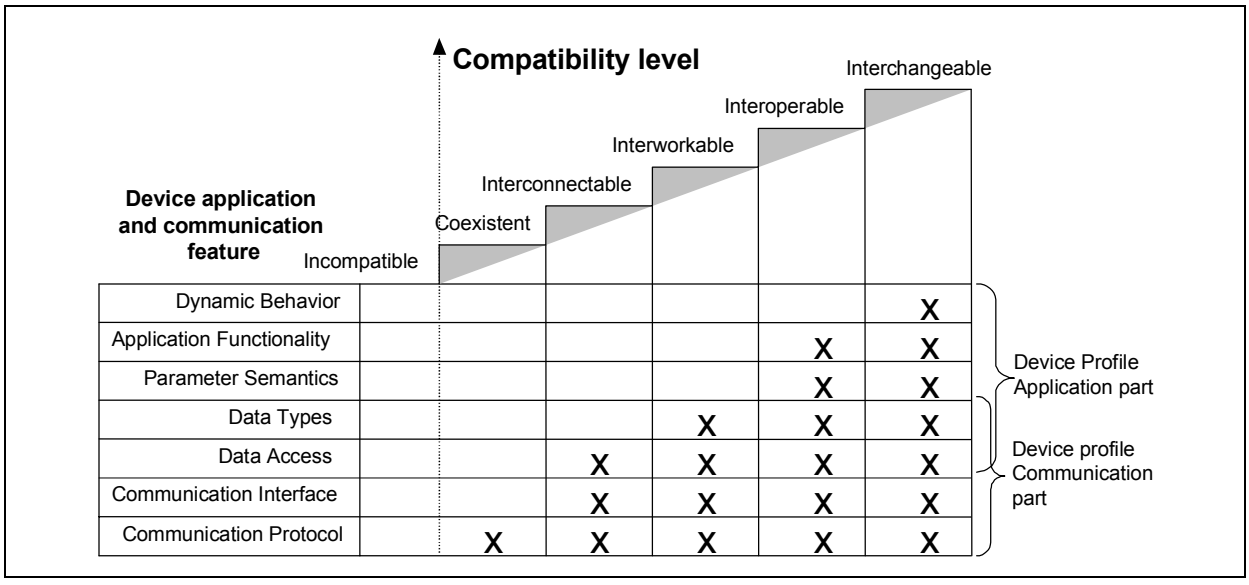

Figure 22 - Levels of functional compatibility (copy of Figure 6)

The descriptions of the compatibility levels are provided in clause 6.2. The device application and communication features are described in Table 3.

Table 3 – Device application and communication features

Feature	Description
Device profile Communication part	
Communication Protocol	This feature is defined by all protocols of layer 1 to 7 of the OSI reference model, i.e. from the physical medium access to the application layer protocol.
Communication Interface	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.
Data access	This feature is defined by the object operation definition or the access parameter attributes of the block data input, data output and parameters
Device profile Application part	
Data types	The data type of the object attributes or block data input, data output or parameter defines this feature.
Data semantics	This feature is defined by the characteristic features (parameter attributes) of the application data this can be data name, data descriptions, the data range, Substitute value of the data, default value, persistence of the data 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 data description part or in a separate behaviour section.
Dynamic performance	This feature is defined by time constraints that influence the data or the general device behaviour. For example the update rate of a process value can influence block algorithms.

The following table shows the relations between device application and communication features and parameter attributes (Table 4).

Table 4 - Relation between parameter attributes and compatibility levels

Parameter attributes	Comm.-Protocol	Comm.-Interface	Data Access	Data Types	Parameter Semantics	Appl. Functionality	Dynamic Behaviour
Parameter Name					X		
Description					X		
ID			X				
Coding					X	X	
Data type				X			
Grouping level				X			
Number of Bytes				X			
Access timing			X				X
Access direction			X				
Persistence						X	
Value Range					X		
Substitute Value						X	
Default Value					X		
Engineering Unit						X	
Offset						X	
Multiplier						X	
Constraint						X	
Supported					X		
Conditional support						X	

The following main features are used for the definition of compatibility levels:

A.3 Data Type

The data types of the device input and output parameters shall be defined in the device profile. The following table shows equivalent some data types used by other standards or consortia. A proposed subset of data types is suggested which is marked as grey fields in the table.

Note: For Time and Date there are currently different formats of the internal coding. IEC 61131-3 requires only a specific format for the external representation.

Table 5 – Data types

Bits	**IEC 61158-5** (Profile Guidelines, Chapter 7.1.1)	**IEC 61131-3** clause 2.3	**COM/DCOM**	**OPC**	**e.g. PNO PA profiles**	**"C"**
1	Boolean *)	BOOL	Variant_BOOL		*)	-
8	-	BYTE	Unsigned char		X	Unsigned char
16	-	WORD -	Unsigned short			Unsigned short
32	-	DWORD - (double)	Unsigned long			Unsigned long
64	-	LWORD (long)	-			-
8	Integer8	SINT (short)	-		X	Char
16	Integer16	INT	short		X	Short
32	Integer32	DINT (double)	long		X	Long
64	-	LINT (long)	-			-
8	Unsigned8	USINT (unsigned short)	-		X	Unsigned char
16	Unsigned16	UINT	-		X	Unsigned short
32	Unsigned32	UDINT	-		X	Unsigned long
64	-	ULINT	-			
32	Floating Point (ANSI/IEEE 754)	REAL (ANSI/IEEE 754)	float		FLOAT (ANSI/IEEE 754)	Float
64	-	LREAL	-			Double
1	-	-	char			
8 x n ASCII	VisibleString (ISO 646)				X	Array of char ?
8 x n	OctetString (ByteString)	STRING (open coding)	BSTR		X	Array of char
-	-	TIME (duration)	-			-
32	TimeDifference (without date) (ms)	-	-		X	-
64	-	DATE	-			-
56	Date [1)] (ms:min:h:day : week:month:years)	-	-		X	-
32	TimeOfDay (without date) (ms - since midnight)	Time_of_day	-		?	-
48	TimeOfDay (with date) (ms since midnight & days since 1.1.1984)		-		?	-

64	-	Date_and_time (open coding)	DATE [3)] (from 1990; from 2000)			-
48	TimeDifference (with date) (ms:days)		-		?	-
	-	ARRAY	SAFEARRAY (Data type)		ARRAY	[ ]
	-	STRUCT	-		RECORD	Struct
	-	Derived data types	-			typedef

A.4 Engineering Unit

English translation and some details to be checked – compare also

http://www.physics.nist.gov/Pubs/SP811/cover.html

Physical dimension	Unit	Abbreviation	Exchange Format ASCII
Length Displacement	Meter Millimeter Kilometer Mikrometer	m mm km µm	m mm km um
Area	Squaremeter Squaremillimeter Squarekilometer	m² mm² km²	m**2 mm**2 km**2
Volume	Cubicmeter Litre	m³ l	M**3 l
Time	Secunde Minute Hour Day Millisecunde Mikrosecunde	s min h d ms µs	s min h d us
Power	Newton Kilonewton Meganewton	N kN MN	N kN MN
Pressure	Pascal Kilopascal Millibar Bar	Pa kPa mbar bar	Pa kPa mbar bar
Mass	Kilogramm Gramm Milligramm Tonne	kg g mg t	Kg g mg t
Energy ?	Joule Kilojoule Megajoule Watt hour Kilowatt hour Megawatt hour	J kJ MJ Wh kWh MWh	J kJ MJ Wh kWh MWh
Real power ?	Watt Kilowatt Megawatt Milliwatt	W kW MW mW	W kW MW mW
Apparent power ?	Voltampere Kilovoltampere Megavoltampere Millivoltampere	VA kVA MVA mVA	VA kVA MVA mVA
Rotation speed ?	1/secunde 1/minute 1/hour	s^{-1} min^{-1} h^{-1}	s**-1 min**-1 h**-1
Angle	Radiant Secunde Minute (Alt-)Grad ? Neugrad (Gon)	rad " ' ° g	Rad “ ‘ ° g
Velocity	Meter/Secunde Millimeter/Sesunde Millimeter/Minute Meter/Minute Kilometer/hour Millimeter/hour Meter/hour Kilometer/hour	m/s mm/s mm/min m/min km/min mm/h m/h km/h	M/s mm/s mm/min m/min km/min mm/h m/h km/h

Physical dimension	Unit	Abbreviation	Exchange Format ASCII
Volume flow ?	Kubikmeter/Secunde Kubikmeter/Minute Kubikmeter/hour Liter/Secunde Liter/Minute Liter/hour	m^3/s m^3/min m^3/h l/s l/min l/h	M**3/s m**3/min m**3/h l/s l/min l/h
Mass flow Massenstrom	Kilogramme/Sekunde Gramme/Sekunde Tonne/Sekunde 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
Drehmoment	Newtonmeter Kilonewton meter Meganewton meter	Nm kNm MNm	Nm kNm MNm
Temperature	Kelvin Grad Celsius Grad 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 Mikroampere	A mA kA µA	A mA kA uA
Electrical Resistance	Ohm Milliohm Kiloohm Megaohm	Ω mΩ kΩ MΩ	O MO KO MO
Verhältnis	Procentage	%	%
relative Feuchte	Procentage	%	%
absolute Feuchte	Gramme/Kilogramme	g/kg	g/kg
Relative Änderung	Percentage	%	%
Frequency	Hertz Kilohertz Megahertz Gigahertz	Hz kHz Mhz GHz	Hz kHz Mhz GHz
Bezogenes Drehmoment ?	Newtonmeter/Ampere	Nm/A	Nm/A
Leistung (US)	Horsepower	H	H
Accelleration	Meter/(s * s)	m/s^2	M/s**2
Jerk	Meter/(s * s * s)	m/s^3	M/s**3

A.5 Object Modelling and UML Syntax

UML class diagram semantic

The class diagram is one part of the UML specification method. For the purpose of the device model the following elements of the class diagram are used as illustrated in Figure 23:

Class
A class is a group of objects with common properties (attributes), common behaviour (methods), common access means to properties and behaviour and the common relations to other objects.

Class attributes
Attribute is a well defined property of an object

Class Operations
Operation is a function of an object which triggers the execution of a method or provide access to an attribute

Inheritance/Specialization
Inheritance defines a relation among classes where one class shares the attributes and/or behaviour of one or more classes.
Specialization provides the ability to create subclasses that represent refinements to the superclass – typically attributes and behaviour are added to the new class

Aggregation
Aggregation provides the capability to create superclasses that consists of several classes.

Association
An Association is a bi-directional semantic between classes

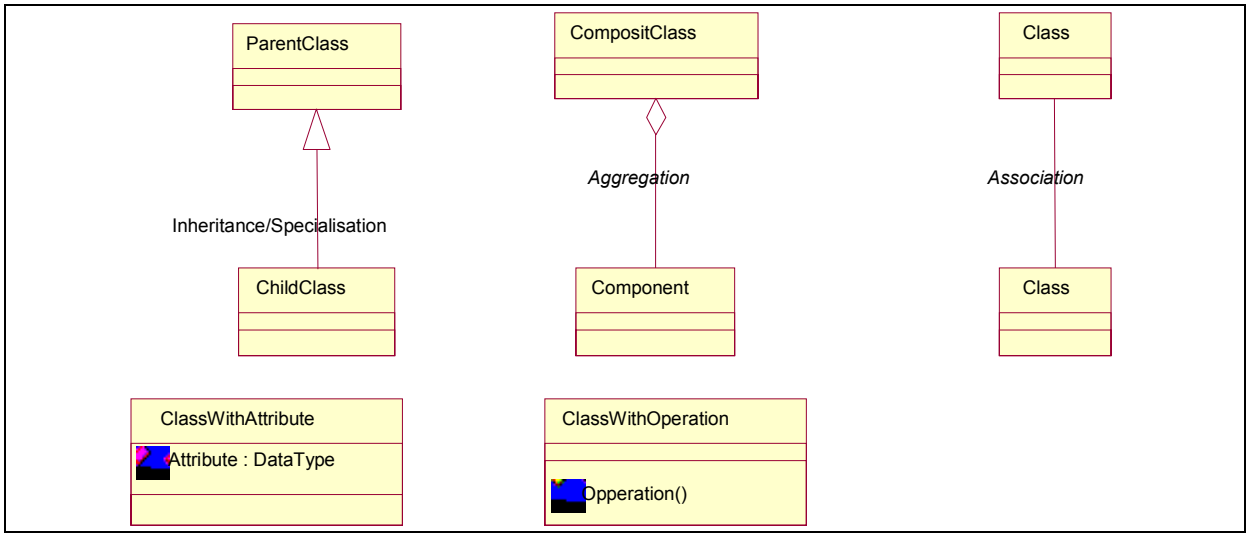

Figure 23 - Description elements of UML class diagrams

A.6 Device Classification examples

The following classification should be a support to locate these devices in the all over industrial automation market. This table is not complete, however the typical field devices of industrial automations are covered.

Table 6 - 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	

A.7 Classification of the algorithms

For each algorithm/method it is suggested to use the following classification:

Table 7 - Algorithm examples

Typical basic functions of devices	Functional Elements
sensor connection sensor range/calibration AD conversion Status estimation	Device Functions
Linearization Filtering Compensation Scaling	Device Functions
Unit Scaling Linearization Simulation	Application Process
Actuation provision Amplification Conversion Status estimation	Device Functions
sensor connection sensor range/calibration AD conversion Status estimation	Device Functions
Scaling Compensation Transition or activity limits	Device Functions
Limit Unit Scaling Linearization Simulation	Application Process
binary control PID mixed discrete and analog calculation	Application Process
Estimation of Device State Test Diagnosis Operating Mode	Device Management

A.8 Collection of parameter attribute

This Table 8 provides a representative collection of parameter attributes as a suggestion which can be used by the profile writer to design the details of the parameter set of the profile template.

Table 8 – Collection of parameter attributes

Parameter attribute	Conceptual Data type	Description
ID	Implementation specific	Identifier of a particular data dictionary entry. This can be an object attribute, a block input, output and parameter. The ID is the key to find a data dictionary entry in the data dictionary. The ID is unambiguous within an object and block.
Parameter Name	String	Name is unique within a block, but may be contained in more than one block. The name is also a key of a data dictionary entry.
Description	String	Semantically meaning of the parameter for which automation engineering terms in AFNOR dictionary, IEC xx Terms and definitions are used.
Coding	String	A: Text; B: Text; ... N: Text; A, B, N are numbers, Text is a string
Data type	Enumeration	IEC 61131 data types have to be preferred.
Grouping level	Enumeration	*Simple* - basic data type *Array* - structured from data of the same type *Record* - structured from data of different data types Any existing record structure shall separately be defined in the profile. Nested data structures are allowed.
Number of Bytes	Numeric	Number of bytes of the data structure.
Access timing	Enumeration	Specification of the dynamic behaviour like • Periodic changes • Episodic changes • Change driven The way to interact within a system is communication system dependent.
Access direction	Enumeration	Specification if a parameter can be read and/or written: • Read only • Read/Write The way to interaction within a system is communication system dependent.
Persistence	Enumeration	Specification whether or not the value is retained in the buffered RAM/EEPROM after the device's power supply was lost. Range of values: • volatile (v) • non-volatile (n)
Value Range	Array of 2 (data type corresponds to the data type parameter attributes)	Definition of the valid range from the lower value to the upper value.
Substitute Value	Corresponds to data type parameter attributes	Defines a substitute value in the application program of a device that must be selected in certain operating states.
Default Value	Corresponds to data type parameter attributes	Defines the value that is contained in the parameter / operation upon delivery. This value is the profile default value. Parameters / operations that do not define a profile default value ("-" character in this column), are delivered with a manufacturer-specific default value.
Engineering Unit	String	Defines the engineering unit (if it has one) of the data dictionary entry
Offset	Numeric	The offset element specifies an offset which is added to an actual value to form an scaled value Engineerng value= (parameter value + offset) * multiplier

Multiplier	Numeric	Scaling factor by which an actual value is multiplied to form a scaled value Engineerng value= (parameter value + offset) * multiplier
Constraint	String	Constraints between parameters, for modelling certain dynamic device behaviour we use relations. Three different relation mechanisms are provided: Disable element, disable another element depending on certain object values. Enable element, enable another element depending on certain object values. Change element, change the value of another element depending on certain object values
Supported	Enumeration	Defines whether or not the data dictionary entry must be supported ("mandatory"). Range of values: • Optional: Data dictionary entry (i.e. parameter) doesn't to be supported. • Mandatory: data dictionary entry (i.e. parameter) shall be supported. • Conditional: Conditional Deployment: This data dictionary entry must be supported if one or more then one another optional data dictionary entry exists. This must be specified under "Conditional Support".
Conditional Support	String	In the case of Supported = Conditional, the data dictionary entry ID of the optional parameter(s) must be specified here. In the case Deployment = True/False, "Empty" shall be entered here.

NOTE One of the main characteristics of a device is its behaviour. This behaviour is hidden in the objects and blocks. The description of one data dictionary entry alone is some times not appropriate. The relationships between more then one data dictionary entry have to be described together. Examples are scaling of measurment or setpoint values, control algorithms, limit checking with hysteresis, calibration. Meeting this requirements an extension shall be introduced in the profile if necessary, compare also subclause 8.2.3.

Conceptual Data Type

The data types used in Table 8 are conceptual types, i.e. they identify the signal type not the implementable data type.

There shall be a mapping to the implementable data types in a certain profile, e.g. to IEC 61131-3 data types as shown in the table annex A.4.