IEC 62714-5

Edition 1.0 2022-03

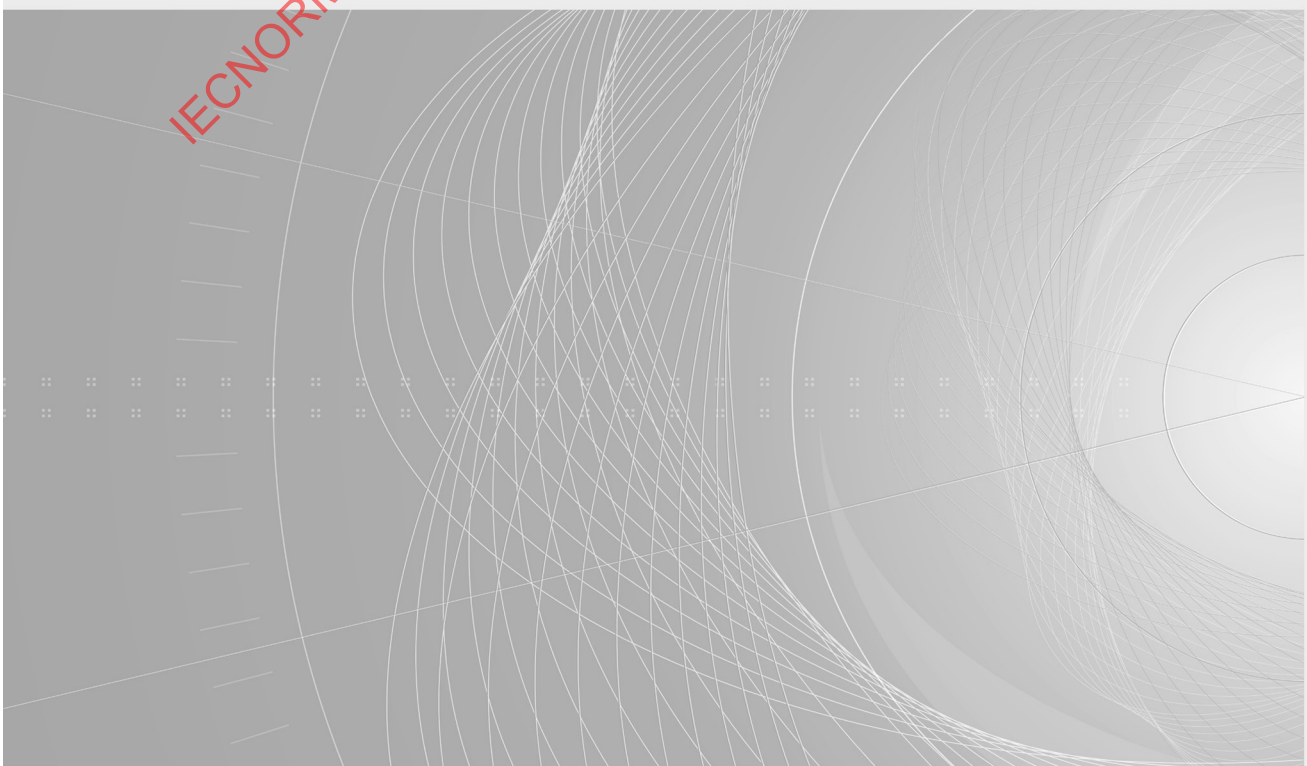
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 5: Communication

Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 5: Communication



IEC 62714-5:2022-03(en-fr)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

IEC 62714-5

Edition 1.0 2022-03

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 5: Communication

Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 5: Communication

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 25.040; 25.040.01

ISBN 978-2-8322-1085-7

CONTENTS

FOREWORD 6
1 Scope 8
2 Normative references 8
3 Terms, definitions, abbreviated terms and acronyms 9
3.1 Terms and definitions 9
3.2 Abbreviated terms and acronyms 9
4 Use cases and network structures 10
4.1 General 10
4.2 Use cases 10
4.2.1 Engineering activities 10
4.2.2 Lossless transfer of communication device instance information 11
4.2.3 Lossless transfer of communication system information 14
4.3 Delimitation of modelling range 16
4.3.1 Scope of the modelling range 16
4.3.2 Interaction structures and life cycles 16
4.3.3 Network objects 17
4.3.4 Network topologies 18
4.3.5 Communication content 23
4.4 Derived modelling requirements 23
5 UML model 24
5.1 Overview 24
5.2 Logical topology 24
5.2.1 Aim of logical topology 24
5.2.2 Item logicalTopology 25
5.2.3 Item logicalConnection 25
5.2.4 Item logicalEndPoint 25
5.3 Physical topology 26
5.3.1 Aim of physical topology 26
5.3.2 Item physicalTopology 26
5.3.3 Item physicalConnection 26
5.3.4 Item physicalEndPoint 27
5.4 Device 27
5.4.1 General 27
5.4.2 Item physicalDevice 27
5.4.3 Item Information 28
5.4.4 Item physicalDeviceInformation 29
5.4.5 Item logicalDeviceInformation 29
5.4.6 Item logicalDevice 29
5.4.7 Item networkDataList 29
5.4.8 Item networkDataItem 29
5.4.9 Item logicalEndPointList 29
5.4.10 Item physicalEndPointList 29
5.4.11 Item physicalChannelList 29
5.4.12 Item physicalChannel 30
5.4.13 Item deviceResource 30
5.4.14 Item variableList 30
5.4.15 Item variable 30

5.4.16 Item pduList 31
5.4.17 Item pdu 31
5.4.18 Item protocolData 32
5.4.19 Item payload 32
5.4.20 Item processDataItemList 32
5.4.21 Item parameterItemList 32
5.4.22 Item dataItem 32
5.4.23 Item processDataItem 32
5.4.24 Item processDataInput 33
5.4.25 Item processDataOutput 33
5.4.26 Item parameterItem 33
6 Representation within AutomationML 33
6.1 Overview of mapping 33
6.1.1 Introduction of mapping 33
6.1.2 General mapping rules 33
6.1.3 Basics 34
6.1.4 Modelling of relations 35
6.1.5 Application process 36
6.2 Basic communication role class library 38
6.2.1 General 38
6.2.2 RoleClass PhysicalDevice 39
6.2.3 RoleClass PhysicalEndpointlist 40
6.2.4 RoleClass PhysicalConnection 40
6.2.5 RoleClass PhysicalNetwork 40
6.2.6 RoleClass LogicalDevice 41
6.2.7 RoleClass LogicalEndpointlist 41
6.2.8 RoleClass LogicalConnection 41
6.2.9 RoleClass LogicalNetwork 41
6.3 Basic communication interface class library 42
6.3.1 General 42
6.3.2 InterfaceClass PhysicalEndPoint 42
6.3.3 InterfaceClass LogicalEndPoint 42
6.4 Steps to model technology specific libraries 43
6.4.1 General 43
6.4.2 Step 1: Development of technology specific role classes 43
6.4.3 Step 2: Development of technology specific interface classes 44
6.4.4 Step 3: Development of system unit class libraries 44
6.4.5 Step 4: Modelling the network 45
6.4.6 Step 5: Modelling the connections 46
6.5 PDU modelling 46
6.5.1 General 46
6.5.2 RoleClass CommunicationPackage 47
6.5.3 InterfaceClass DatagrammObject 48
6.5.4 Steps to model technology specific libraries 49
6.6 References to attributes 51
6.7 Usage of metadata 53
Bibliography 55

Figure 1 – General engineering activities communication system engineering is embedded within 10
Figure 2 – Information flow of the use case 12
Figure 3 – Alternative information flow of the use case 13
Figure 4 – Information flow of the use case 15
Figure 5 – Example of a logical level view on communication systems 17
Figure 6 – Example of a physical level view on communication systems 18
Figure 7 – Combined views on communication systems 18
Figure 8 – Star topology example 19
Figure 9 – Ring topology example 19
Figure 10 – Line topology example 20
Figure 11 – Simple network with direct wiring 20
Figure 12 – Network with active infrastructure 21
Figure 13 – Networks connected by gateways 21
Figure 14 – Hierarchical structured networks 22
Figure 15 – Network covering multiple applications 22
Figure 16 – General modelling strategy for PDUs 23
Figure 17 – Structure of communication network 24
Figure 18 – View on logical topology 25
Figure 19 – View on physical topology 26
Figure 20 – Part 1 of the device model 28
Figure 21 – Part 2 of the device model 31
Figure 22 – Communication role class library and communication interface class library 35
Figure 23 – Derived role class libraries and interface class libraries for a special example 35
Figure 24 – SystemUnitClassLib examples for communication system modelling 37
Figure 25 – Final network model example 38
Figure 26 – Basic communication role class library 39
Figure 27 – CommunicationRoleClassLib 39
Figure 28 – XML text of the communication role class library 39
Figure 29 – Basic communication interface class library 42
Figure 30 – CommunicationInterfaceClassLib 42
Figure 31 – XML text of the communication interface class library 42
Figure 32 – Derivation of a technology specific role class library out of the base role class library 43
Figure 33 – Derivation of a technology specific role class library out of the base role class library 44
Figure 34 – Technology specific <SystemUnitClassLib>s 45
Figure 35 – Technology specific communication network 46
Figure 36 – Extended communication role class library 47
Figure 37 – Extended CommunicationRoleClassLib 47
Figure 38 – XML text of the extended communication role class library 47
Figure 39 – Extended communication interface class library 48
Figure 40 – Extended CommunicationInterfaceClassLib 48

Figure 41 – XML text of the extended communication role class library 48
Figure 42 – Derivation of a technology specific role class library out of the extended role class library 49
Figure 43 – Derivation of a technology specific interface class library out of the extended interface class library 50
Figure 44 – Technology specific extended <SystemUnitClassLib>s 50
Figure 45 – Technology specific communication network with communication package models 51
Figure 46 − Field SourceDocumentInformation according to communication related libraries 54

Table 1 – Mapping rules 34
Table 2 – Modelling of relations in AutomationML 36
Table 3 – RoleClass PhysicalDevice 40
Table 4 – RoleClass PhysicalEndpointlist 40
Table 5 – RoleClass PhysicalConnection 40
Table 6 – RoleClass PhysicalNetwork 40
Table 7 – RoleClass LogicalDevice 41
Table 8 – RoleClass LogicalEndpointlist 41
Table 9 – RoleClass LogicalConnection 41
Table 10 – RoleClass LogicalNetwork 41
Table 11 – InterfaceClass PhysicalEndPoint 42
Table 12 – InterfaceClass LogicalEndPoint 43
Table 13 – RoleClass CommunicationPackage 48
Table 14 – InterfaceClass DatagrammObject 49
Table 15 – Communication related attributes 52

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

ENGINEERING DATA EXCHANGE FORMAT FOR USE IN INDUSTRIAL AUTOMATION SYSTEMS ENGINEERING – AUTOMATION MARKUP LANGUAGE –

Part 5: Communication

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 62714-5 has been prepared by subcommittee 65E: Devices and integration in enterprise systems, of IEC technical committee 65: Industrial-process measurement, control and automation. It is an International Standard.

The text of this International Standard is based on the following documents:

Draft	Report on voting
65E/844/FDIS	65E/886/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/standardsdev/publications.

A list of all parts of the IEC 62714 series, under the general title *Engineering data exchange format for use in industrial automation systems engineering – Automation markup language,* can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

ENGINEERING DATA EXCHANGE FORMAT FOR USE IN INDUSTRIAL AUTOMATION SYSTEMS ENGINEERING – AUTOMATION MARKUP LANGUAGE –

Part 5: Communication

1 Scope

Engineering processes of technical systems and their embedded automation systems are executed with increasing efficiency and quality. Especially since the project duration tends to increase as the complexity of the engineered system increases. To solve this problem, the engineering process is more often being executed by exploiting software based engineering tools exchanging engineering information and artefacts along the engineering process related tool chain.

Communication systems establish an important part of modern technical systems and, especially, of automation systems embedded within them. Following the increasing decentralisation of automation systems and the application of fieldbus and Ethernet technology connecting automation devices and further interacting entities need to fulfil special requirements on communication quality, safety and security. Thus, within the engineering process of modern technical systems, engineering information and artefacts relating to communication systems also need to be exchanged along the engineering process tool chain.

In each phase of the engineering process of technical systems, communication system related information can be created which can be consumed in later engineering phases. A typical application case is the creation of configuration information for communication components of automation devices including communication addresses and communication package structuring within controller programming devices during the control programming phase and its use in a device configuration tool. Another typical application case is the transmission of communication device configurations to virtual commissioning tools, to documentation tools, or to diagnosis tools.

At present, the consistent and lossless transfer of communication system engineering information along the complete engineering chain of technical systems is unsolved. While user organisations and companies have provided data exchange formats for parts of the relevant information like FDCML, EDDL, and GSD, the above named application cases cannot be covered by a data exchange format. Notably the networking related information describing communication relations and their properties and qualities cannot be modelled by a data exchange format.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

IEC 61131-10:2019, *Programmable controllers – Part 10: PLC open XML exchange format*

IEC 62424:2016, *Representation of process control engineering − Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools*

IEC 62714-1, *Engineering data exchange format for use in industrial systems engineering – Automation Markup Language – Part 1: Architecture and general requirements*

IEC 62714-4, *Engineering data exchange format for use in industrial systems engineering – Automation markup language − Part 4: Logic*

IEC 81346 (all parts), *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations*

3 Terms, definitions, abbreviated terms and acronyms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 62714-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
AutomationML/AML
XML based data exchange format for plant engineering data

3.1.2
Automation object
entity in an automated system

Note 1 to entry: An example of an automation object is an automation component, a valve, or a signal.

3.2 Abbreviated terms and acronyms

For the purposes of this document, the following abbreviations apply.

AML	Automation Markup Language
CAEX	Computer Aided Engineering Exchange as defined in IEC 62424:2016
ECAD	Computer aided engineering for electrical engineering
EDDL	Electronic Device Description Language
FDCML	Field Device Configuration Markup Language
GUID	Global Unique Identifier
GSD	General Station Description
HMI	Human Machine Interface
ID	Identifier
MCAD	Computer aided engineering for mechanical engineering
OPC	Open Platform Communications
PDU	Protocol Data Unit
SCADA	Supervisory Control And Data Acquisition
UML	Unified Modelling Language
UUID	Universal Unique Identifier
XML	Extensible Markup Language

4 Use cases and network structures

4.1 General

The modelling of communication systems based on AutomationML targets the modelling of a large amount of information created, exchanged, and applied within the engineering process of manufacturing systems. Nevertheless, not all possible communication system related information will be modelled. Within the following subclause, the use cases for the application of the communication system modelling as well as the relevant information sets within them are named.

4.2 Use cases

4.2.1 Engineering activities

Network topology and communication system related information is relevant to various engineering activities along the engineering chain of production systems. Within the engineering process of production systems, the communication system can be designed in the detailed engineering phase exploiting various tools. Thereby, communication system related information is created which subsequently can be applied within the detailed design of devices and the device commissioning. Figure 1 represents an example set of engineering activities relevant within the general engineering process of production systems and the communication system engineering embedded within.

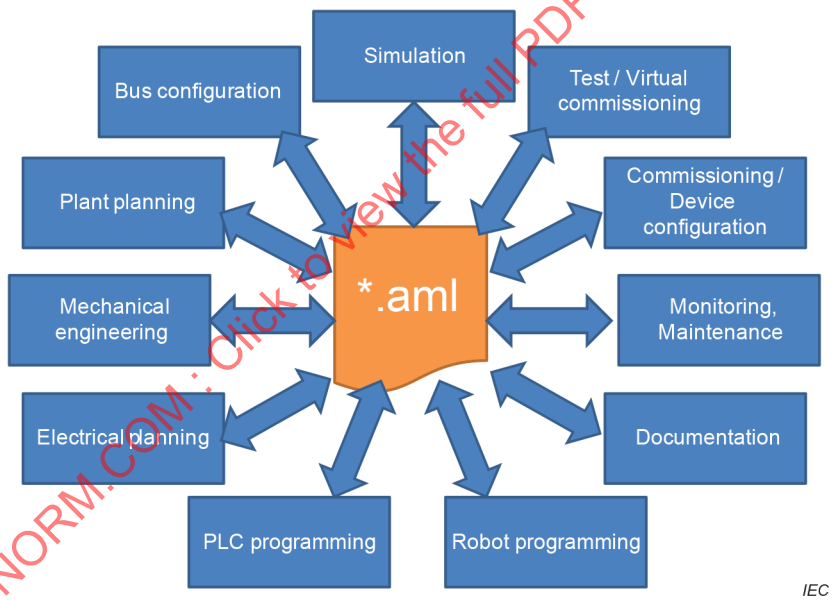


Figure 1 – General engineering activities communication system engineering is embedded within

Within the named engineering activities, engineering tools like (but not limited to) the following will have a relevant impact:

- plant planning tools,
- mechanical engineering tools (MCAD),
- electrical engineering tools (ECAD),
- controller programming tools,
- robot programming tools,
- HMI programming tools,
- OPC system configuration tools,

- device configuration tools,
- bus configuration tools,
- simulation tools,
- SCADA systems,
- virtual commissioning tools,
- documentation tools,
- communication system security tools,
- communication system management tools,
- communication system diagnosis tools.

These tools will create and/or consume communication system related engineering information depending on the use case of the engineering chain.

Nevertheless, (among others) there are two main application cases within this engineering chain, where communication system related information can be exchanged between engineering tools. These two use cases are the main target of the modelling of communication systems based on AutomationML.

4.2.2 Lossless transfer of communication device instance information

Within the general engineering process, this use case covers the transition of communication relevant information for configuration of communication components of sensors and actuators from controller programming tool and similar tools to sensor and actuator configuration/programming tools. It contains the transition of information relevant for correct communication (like addresses and channels) as well as for correct structuring of communication data packages transmitted within communication (like transmitted data points of sensors). An overview is shown in Figure 2.

Within the related engineering activities, engineers with the engineering roles of controller programmer, HMI programmer, electrical design engineer, commissioner communication, commissioner controller, and robot programmer can be involved. They will execute the following sequence of engineering and data exchange activities which should be seen as an example sequence.

Step 1. Design of system instrumentation, definition of used/interconnected devices

Step 2. Export of device information from system instrumentation tool

Step 3. Import of device information to controller programming and device configuration tools

Step 4. Integration of device descriptions (like GSD) in controller programming tool

Step 5. Design of controller programs and configurations within controller programming tool

Step 6. Export of communication device relevant information from controller programming tool

Step 7. Import of communication device relevant information to device configuration tool

Step 8. Use of information for parameterisation of communication component of device (addresses, etc.) and for structuring of communication packages (send data points)

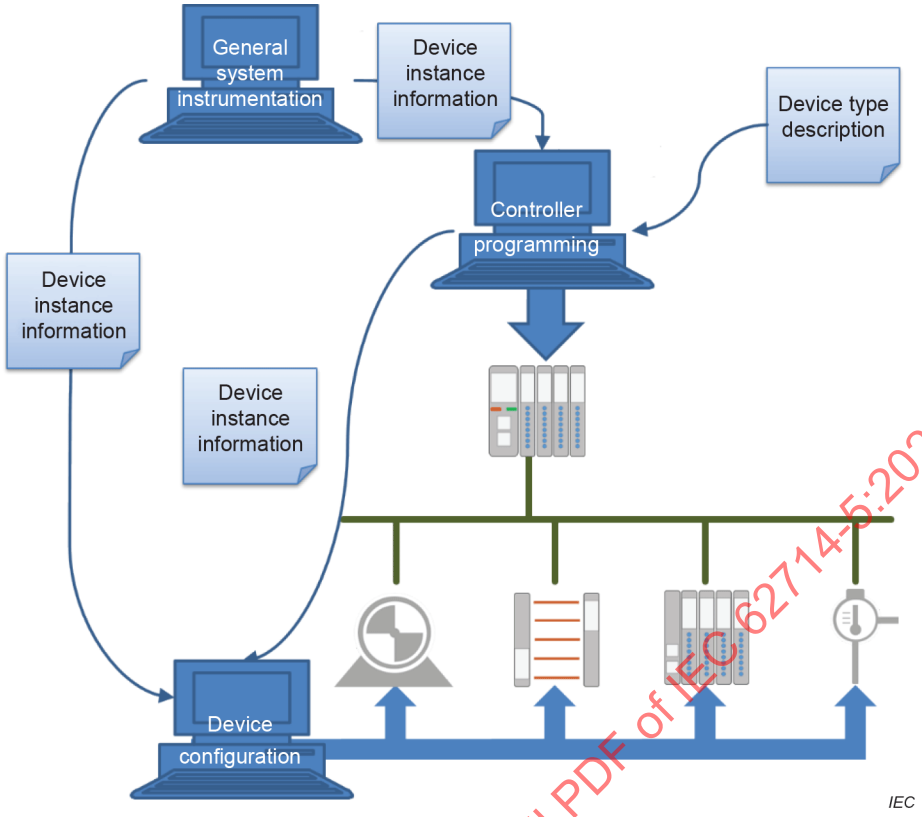


Figure 2 – Information flow of the use case

There are different alternatives possible but not usually applied. The following two sequences are imaginable which will be in the focus of the use case.

Sequence 1:

Step 1. Design of communication system configurations within third party tool

Step 2. Integration of device descriptions

Step 3. Export of communication device relevant information

Step 4. Import of communication device relevant information to device configuration/programming tool

Step 5. Use of information for parameterization of communication component of device (addresses, etc.) and for structuring of communication packages (send data points)

Sequence 2:

Step 1. Third party tool, e.g. the device configuration tool provides information like signals, data volume, describing the instance information

Step 2. Device vendor provides device descriptions, describing the type information

Step 3. Bus configuration tool consumes these snippets and device descriptions

Step 4. Bus configurator generates bus configuration

Step 5. Import of the bus configuration into the controller programming tool

Step 6. Use of information from peripheral devices inside the controller program

Both sequences are commonly depicted in Figure 3.

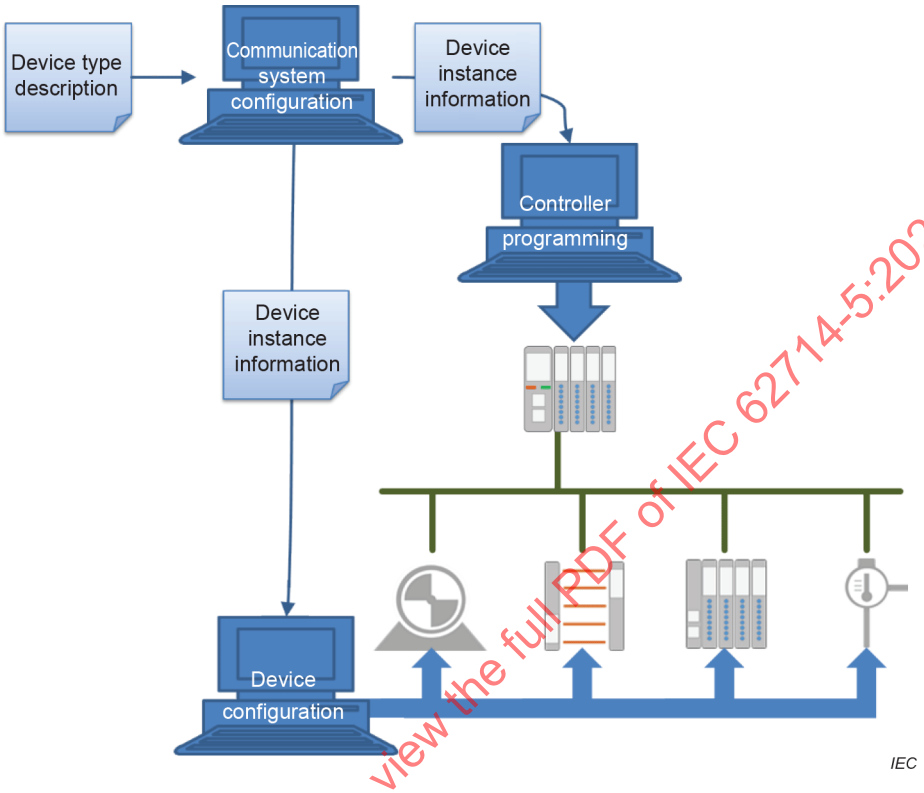


Figure 3 – Alternative information flow of the use case

Possible tools exporting communication system information can be e.g. controller programming tools. They cover controller programming projects with data points (variables), device configurations, and indirect communication network descriptions. In addition, communication system engineering tools can be data sources as well as instrumentation tools like ECAD tools.

Data sinks of the data exchange can be tools for sensor communication configuration, HMI programming, robot programming, OPC system configuration, or actor communication configuration. They mostly cover programming projects with data points (variables), device configurations, and indirect communication network descriptions. Relevant tools can be FDT tools, HMI programming tools, and robot programming tools.

Based on this use case, the modelling of communication systems based on AutomationML should cover information about IO lists, association of variables (IOs) to communication data packages and device parameters for parameterization of communication devices such as addresses (e.g. IP address), media access (e.g. MAC address), subnet masks, gateway addresses, and communication objects used, such as profile information.

In addition, the modelling should fulfil the following non-functional requirements. Device parameter list shall be extendable by users to cover upcoming technologies. Appropriate RoleClassLibs and/or SystemUnitClassLibs enabling the identification of object semantics shall be defined.

4.2.3 Lossless transfer of communication system information

Beyond the configuration of communication devices, communication system information is exploited in different engineering, monitoring, maintenance, etc. tools. Therefore, this use case covers the transmission of communication network configuration and structure information including infrastructure device configuration, end device configuration with respect to communication system parameters, network structure and wiring, quality of service, etc. This information can be provided to device configuration tools, documentation and maintenance tools, and network management tools. They can enable the combination of physical wiring with logical communication connections for error detection.

Within this use case, engineers with the engineering roles controller programmer, HMI programmer, electrical design engineer, commissioner communication, commissioner controller, robot programmer, and operator are involved. They will execute the following engineering process which should be seen as an example sequence.

Step 1. Engineering of sets of variables to be transmitted within communication system within controller programming tools and sensor/actuator programming/configuration tools

Step 2. Export of these variable sets from controller programming tools and sensor/actuator programming/configuration tools

Step 3. Import of variable sets to communication network engineering tools

Step 4. Engineering of mapping lists of variables and engineering of physical wiring

Step 5. Export of communication system design from communication network engineering tools

Step 6. Import of communication system design to continuing engineering or execution tools

The sequence is commonly depicted in Figure 4.

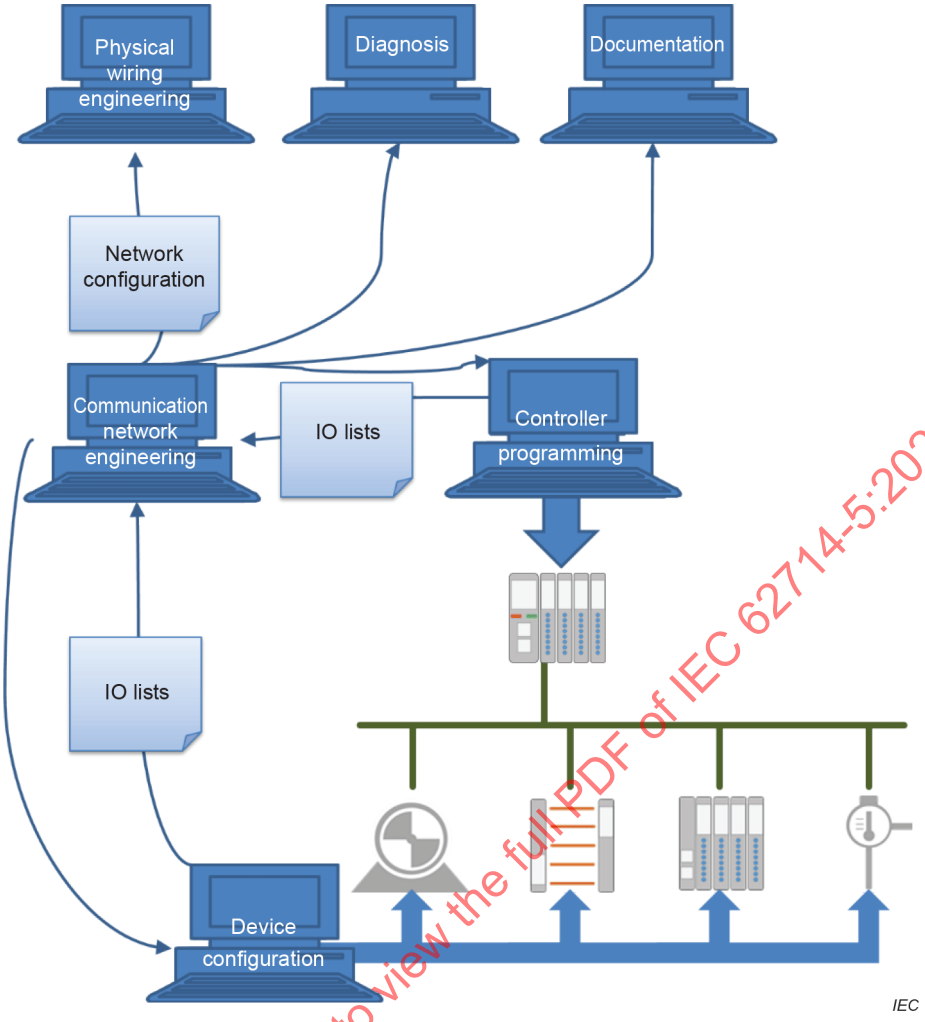


Figure 4 – Information flow of the use case

Source of the engineering data exchanged can be controller programming tools covering controller programming projects with data points (variables), device configurations, and indirect communication network descriptions, communication system engineering tools covering communication system projects including device configurations and communication connection properties, sensor communication configuration tools, HMI programming tools, robot programming tools, OPC system configuration tools, or actor communication configuration tools all together covering programming projects with data points (variables), device configurations, and indirect communication network descriptions.

Sinks of the data exchange can be tools for communication system security monitoring, communication system configuration, communication system management, and communication system diagnosis.

Based on this use case, the modelling of communication systems based on AutomationML shall cover information about network topology, connector lists of devices and their properties, physical connections between connectors of devices and their properties, logical links between device based applications and their properties, mapping of logical links to physical connections, association of link and/or connection attributes to links or connections like sender and receiver addresses, quality of service, protocols used, etc.

In addition, some non-functional requirements are derived. Link and connection attribute list shall be extendable by users to cover upcoming technologies. Appropriate RoleClassLibs and/or SystemUnitClassLibs enabling the identification of object semantics shall be defined.

4.3 Delimitation of modelling range

4.3.1 Scope of the modelling range

Within the scope of the use cases named above, different communication technologies, communication system use strategies and life cycles as well as communication network architectures can be considered. Within the following, the broad theoretical range will be limited to most applied cases.

4.3.2 Interaction structures and life cycles

Following the above described use cases, the communication system modelling based on AutomationML should concentrate on the description of cyclic communication based on a fixed assignment between communication partners. These communication partners will run (more or less complex) parts of the control logic of control applications on different communicating control devices.

Therefore, there are at least two different views relevant to the communication system; a logical view and a physical view. On a high level logical view, there are data processing applications exchanging application data (variables) logically. For the physical realisation of the data processing applications control devices are used. These devices will physically communicate using communication systems. Within both views, the modelled objects (application parts, devices, connections, etc.) will have special properties which are relevant for the correct use/behaviour of the communication system. Such systems and their descriptive information are in the scope of the following presentations.

Out of scope of the communication system modelling based on AutomationML are for example

- acyclic service based and only temporarily used communication structures as they are usually present in the use of the World Wide Web,
- safety or security related issues, and
- complex device behaviour and structure information as they are usually covered by device description languages.

Nevertheless, the use of AutomationML to model communication systems with these properties is not excluded although not intended.

Communication system descriptions can cover different levels of detail. They can range from very abstract descriptions only naming communicating partners and their dependencies down to a detailed description of communication connections and their properties as well as device and application parts with their properties. These different levels of detail are in the scope of the intended communication system modelling.

Communication systems are complex systems covering different technologies within one communication system. As an example, controller based control systems can be considered. They usually embed different fieldbus systems, Ethernet based communication, copper and fibre optic wiring, wireless communication, different active infrastructure, etc. Different communication protocols can operate on the same physical infrastructure (wire, wireless channel etc.). A single device may execute several control application parts. Thus, the different communication technologies are cross-linked and depend on each other. This complexity and interlinking should be expressible by AutomationML and is in scope of the following descriptions.

Communication systems are considered in different phases of the engineering of a production system such as detailed engineering and commissioning and are used within different phases of the life cycle of a production system like device configuration, production ramp up, normal production run, diagnosis and maintenance. These different phases of engineering and use of production systems and their impact on model content and model use should be covered by the AutomationML based modelling of communication systems.

4.3.3 Network objects

Each communication network will be seen as two layered: a logical level and a physical level. Mapping this view on the often referenced ISO OSI 7 layer structure of communication systems [2][1], the physical level will cover the layers 1 and 2 of the ISO OSI 7 layer structure while the logical level subsumes layers 3 to 7 of the ISO OSI 7 layer structure and the control applications above layer 7.

Looking on the logical layer we can find the following entities to be modelled. Control applications usually consist of multiple distributed application parts providing different functionalities of the control process and forming logical devices. An example (given in Figure 5) can be a main control application exchanging sensor and actuator information with two IO functions. In general, these logical devices (control application parts) exchange information of different types, in our example different sensor signals and actuator states, which can be seen as connection points to the logical devices and end points of the information exchange between logical devices. The information exchange itself is executed by different logical connections.

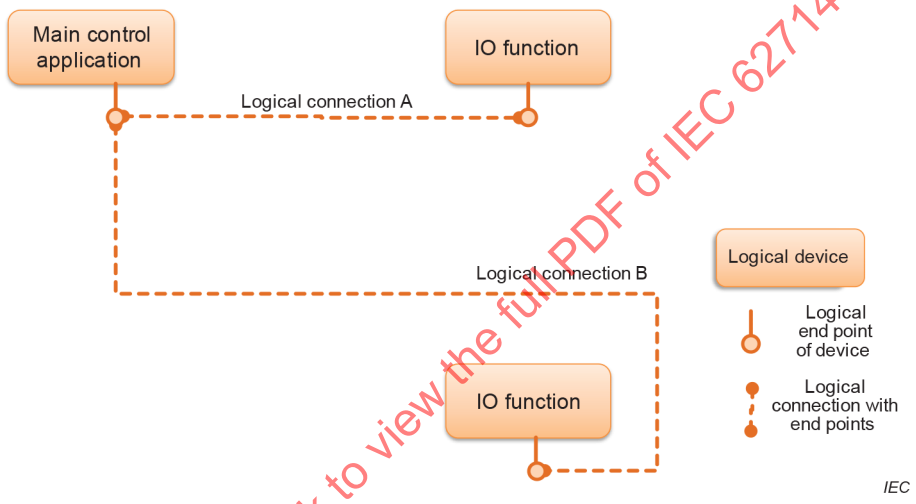


Figure 5 – Example of a logical level view on communication systems

The logical network may contain different objects with different descriptive properties. While logical control devices may have unique identifiers, cycle times, and foot print (to name only a few examples), logical end points may have a data type, and logical connections may have a required transmission rate. In any case, these descriptive properties can be considered as attributes of the objects of interest.

On the physical level, physical devices with physical interfaces can be found. In the example given in Figure 6, a controller is connected via an active infrastructure with two IO devices. Physical devices have physical endpoints representing network interfaces like plugs and sockets and are connected via physical connections to a communication system. Compared to the logical level view, there are additional physical entities representing infrastructure components of the network (like switches, etc.).

As in the case of the logical network, the physical network objects may have different descriptive properties. While physical devices may have identifiers and processing capabilities, physical end points may have an address, a maximal data rate and a connector type, whereas physical connections may have a possible transmission rate, a wire type, antenna type, plug type or a documentation number. These descriptive properties can be considered as attributes of the objects of interest.

[1] Figures in square brackets refer to the Bibliography.

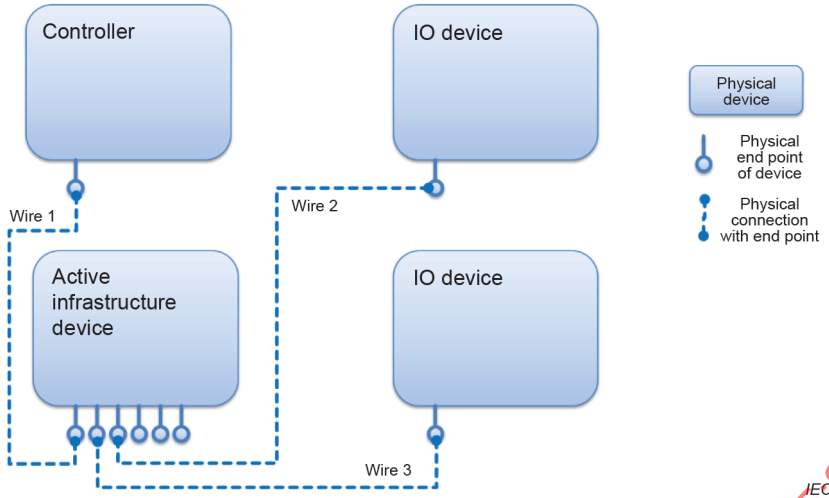


Figure 6 – Example of a physical level view on communication systems

Both views need to be combined to get the complete communication system description. Therefore, logical devices are hosted by physical devices interconnected in a physical network topology. In the example given in Figure 7, a controller hosts a main control application while the IO devices host IO functions. In addition, physical interfaces and logical interfaces are mapped. Thus, each logical connection is mapped virtually to a set of physical connections implementing it. It is not strictly necessary that there is a unique chain of physical connections representing this implementation since not all communication systems support this.

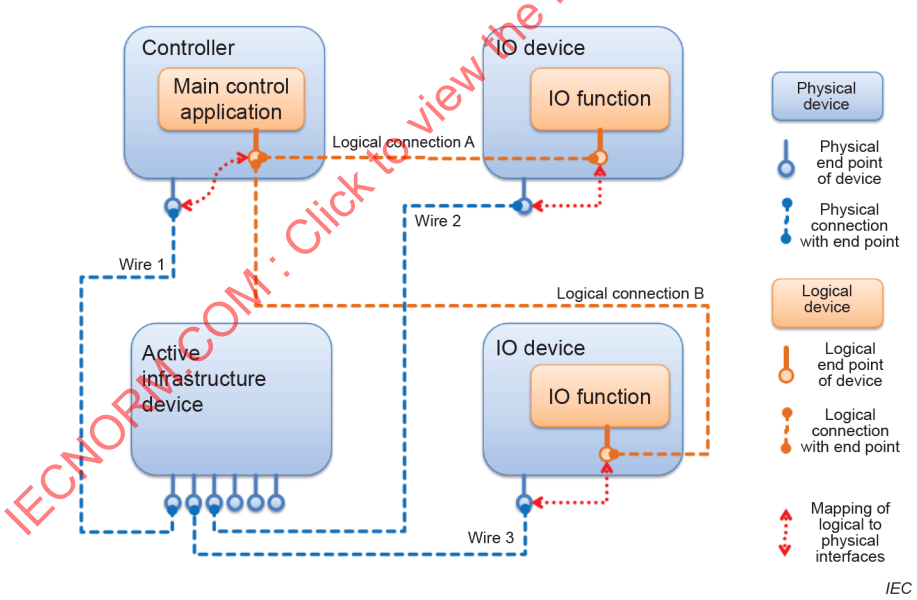


Figure 7 – Combined views on communication systems

4.3.4 Network topologies

Within industrial communication systems, different network architectures are to be considered. On the one hand different wiring topologies can be distinguished. On the other hand different physical network topologies are possible, often dependent on cable routing or wireless channel settings. The communication system modelling presented here should be applicable to all of them.

With respect to different wiring topologies, the modelling should be able to cover star, line, and ring topologies including their combinations. Some examples are given in Figure 8, Figure 9, and Figure 10.

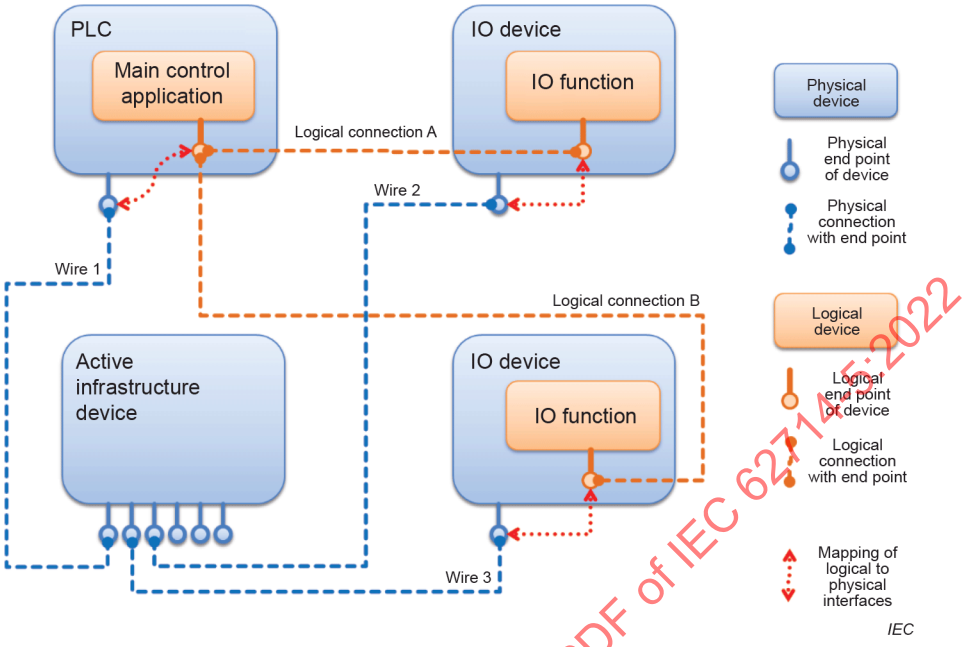


Figure 8 – Star topology example

Figure 8 presents a typical star topology where the active infrastructure device establishes connections between the different physical devices while the logical connections remain bilateral connections between logical devices.

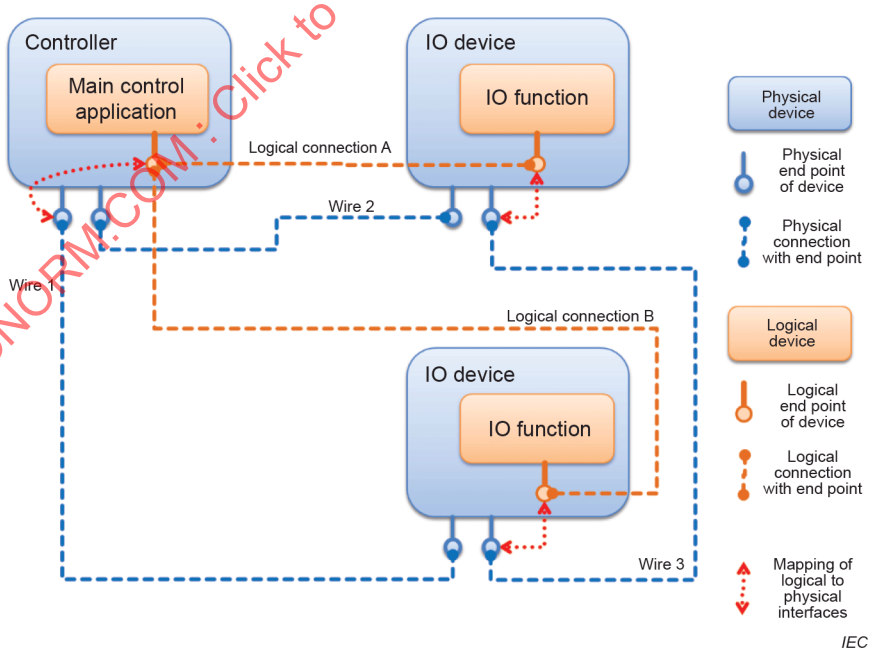


Figure 9 – Ring topology example

Figure 9 presents a typical ring topology where each physical device has two physical interfaces connecting the devices in a physical ring. Again, the logical connections remain bilateral connections between logical devices.

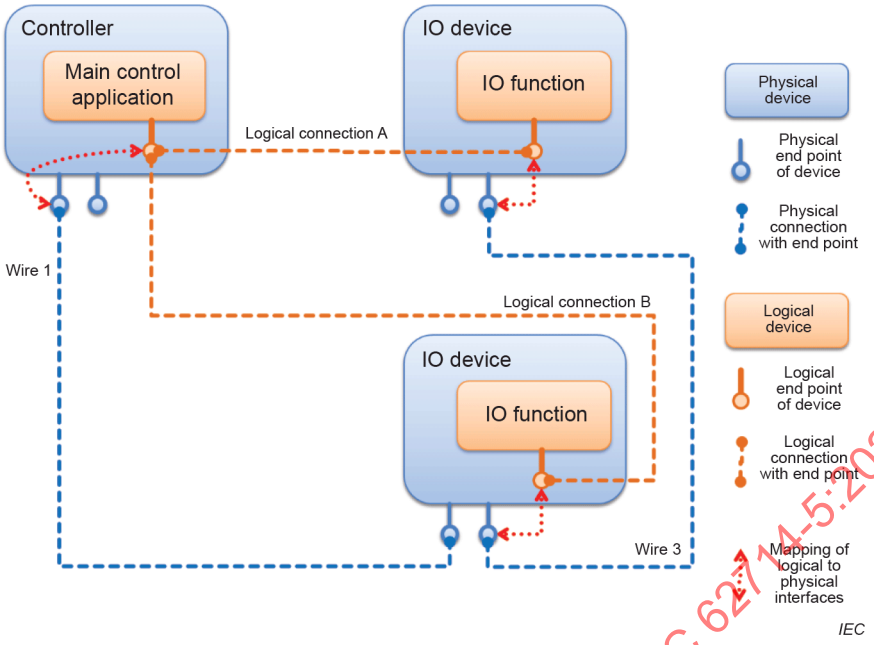


Figure 10 – Line topology example

Figure 10 presents a typical line topology. Here again each physical device has two physical interfaces connecting the devices in a physical line. Thus, it is an open ring topology. The logical connections remain bilateral connections between logical devices.

With respect to network topologies, the modelling should enable the representation of simple networks of direct wiring of components, networks with active infrastructures, networks connected by gateways and hierarchical networks (i.e. networks, where one net component contains a complete further network like a backplane bus in a modular controller). Examples of such network structures are given in Figure 11, Figure 12, Figure 13, and Figure 14.

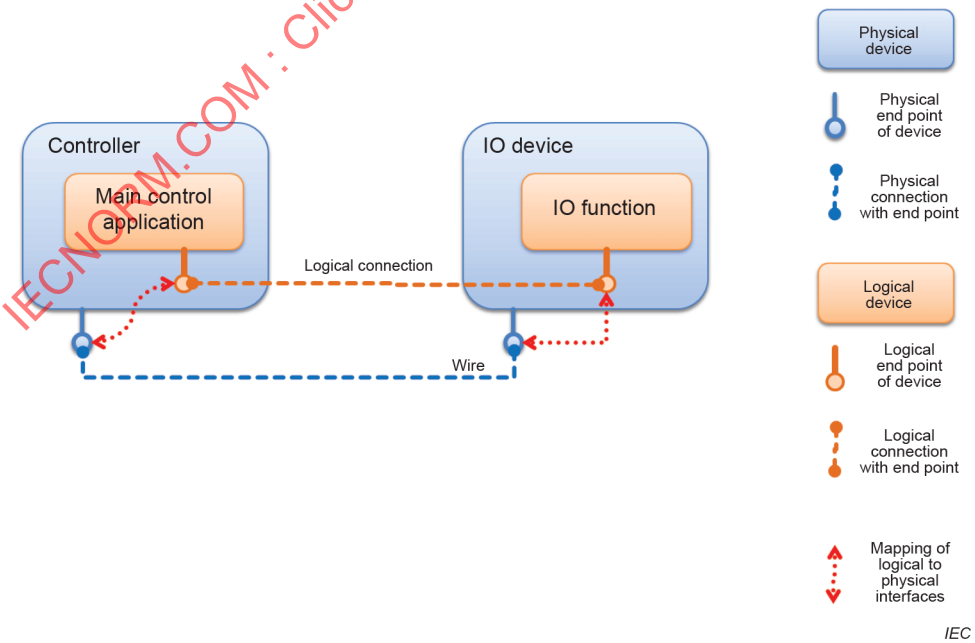


Figure 11 – Simple network with direct wiring

Figure 11 represents a very basic network where two physical devices are directly wired establishing a logical connection between the relevant logical devices. It can be seen as the minimal network which shall be representable.

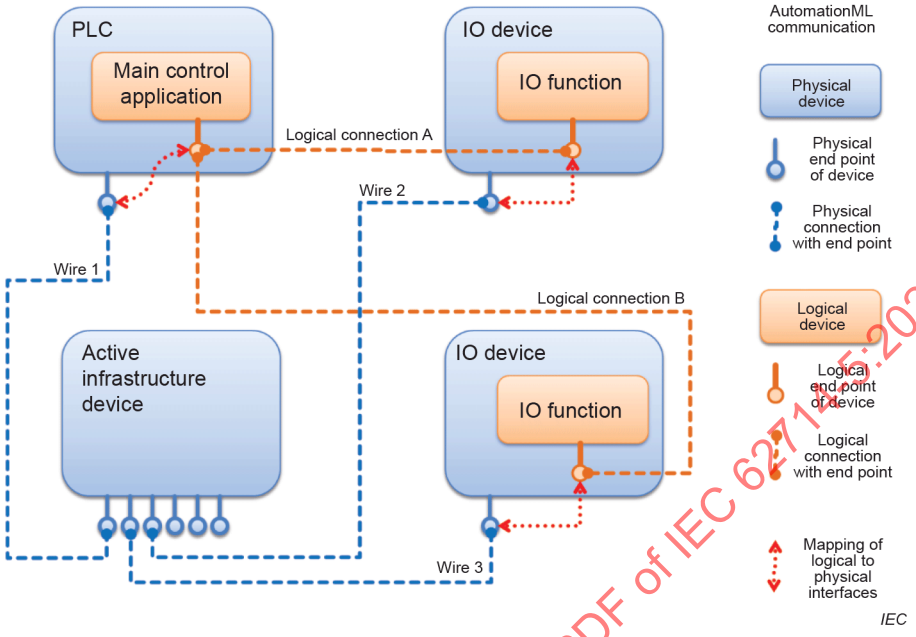


Figure 12 – Network with active infrastructure

Figure 12 represents a network containing active infrastructure devices as is usually the case in Ethernet based networks where switches, hubs, and routers are integrated to forward communication.

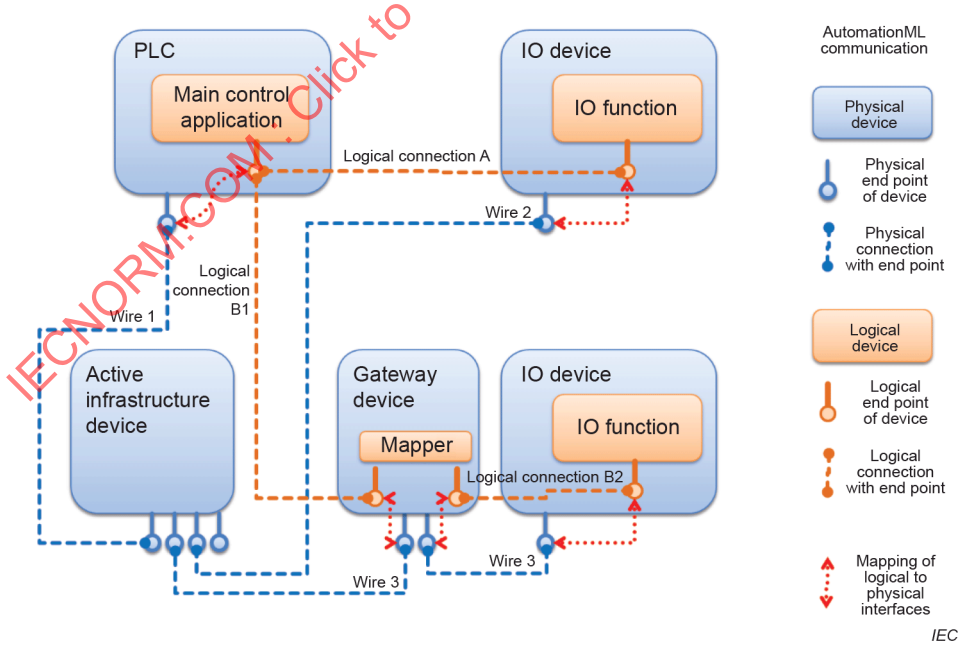


Figure 13 – Networks connected by gateways

Figure 13 represents a network where a logical connection spans two different networks exploiting an application gateway for mapping of logical connections. It depicts that within this gateway, the data items of the two relevant connections are mapped to each other.

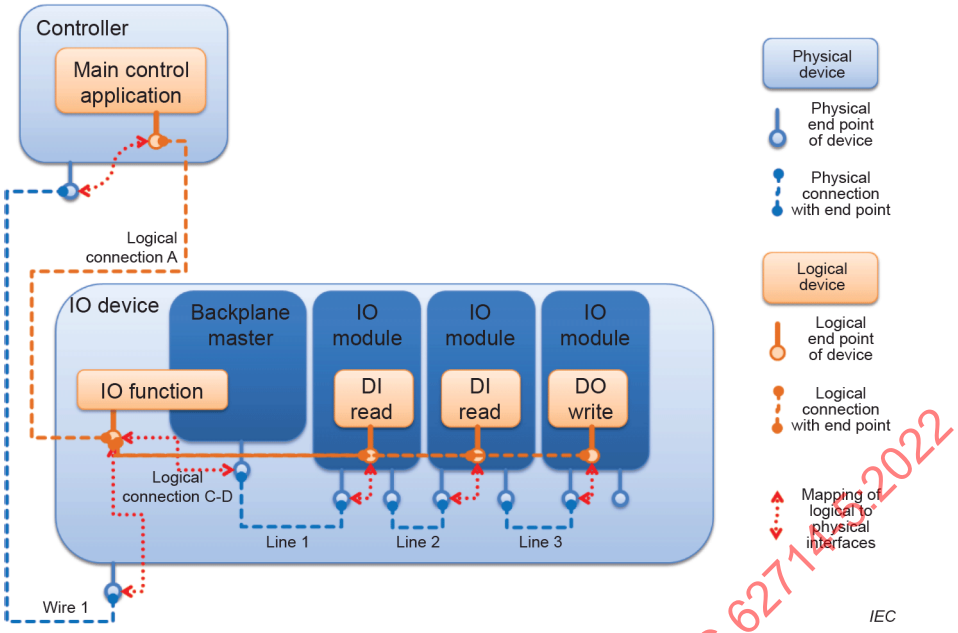


Figure 14 – Hierarchical structured networks

Figure 14 represents a hierarchical network structure as usually given in the case of a communication system containing a physical device with a backplane network connecting the different internal components of the physical device. This case is given if modular fieldbus couplers are exploited in a network.

In addition to the possible application of different topologies and network interaction structures within industrial communication systems, there are usually different applications connected by a communication system like control and system configuration/programming. Networks covering multiple applications should also be modelled as shown in Figure 15.

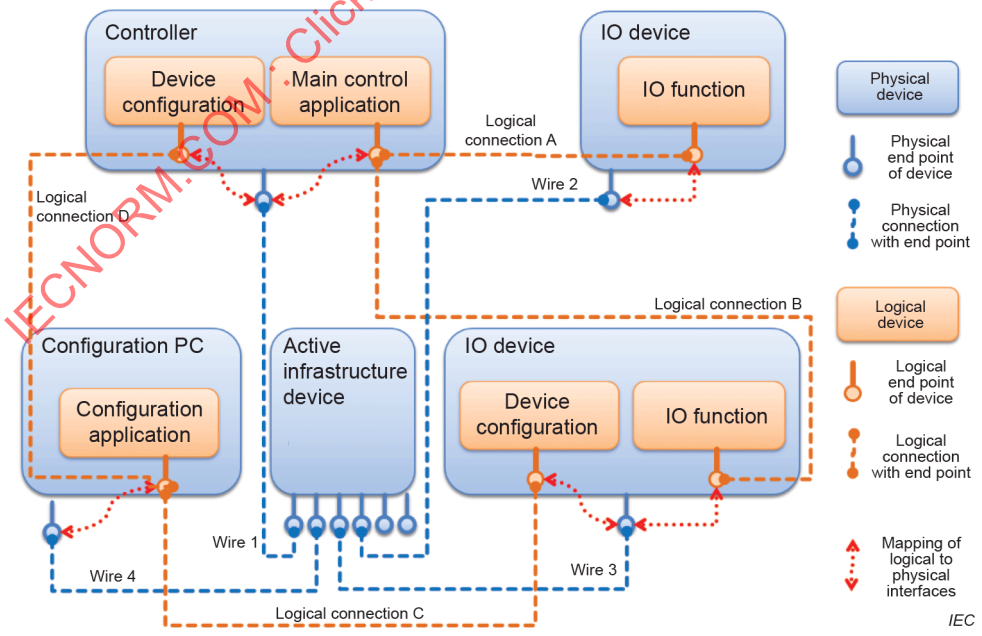


Figure 15 – Network covering multiple applications

4.3.5 Communication content

Within communication systems, communication datagrams (known as Protocol Data Units/ PDUs) are exchanged between communication partners, i.e. control application parts. Therefore, each logical connection contains PDU objects exchanged via this connection. In the case of a PDU object containing control information (sensor and actor signals, status, alarms, etc.), it may be linked to a PLCopenXMLInterface (see IEC 61131-10) or a SignalInterface modelling the exchanged information.

This structure is depicted in Figure 16.

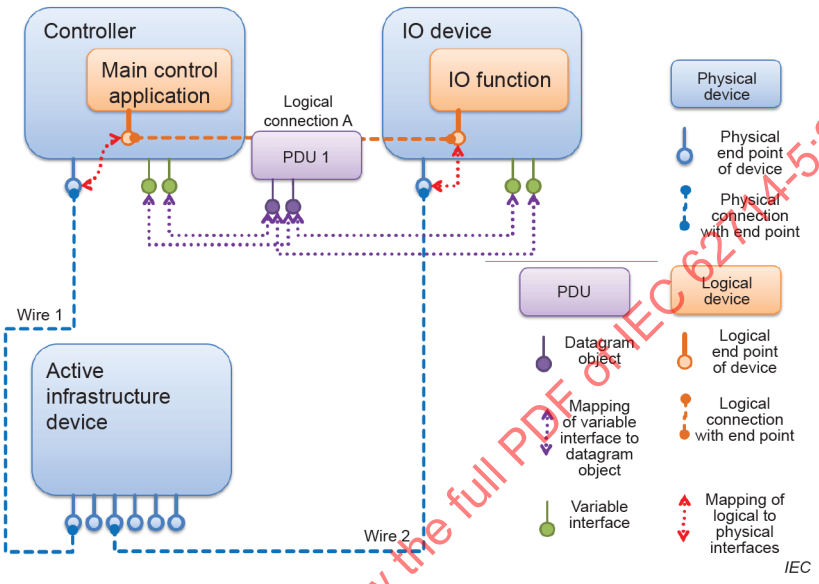


Figure 16 – General modelling strategy for PDUs

4.4 Derived modelling requirements

Based on the observations given in this subclause, the following general information is modelled:

a) Physical level information
   1) Physical devices representing the physically communicating entities with their properties,
   2) End points of physical connections attached to physical devices representing the physical interface of the devices (socket) with their descriptive properties,
   3) Physical connections representing the physical wiring or used wireless channels with their descriptive properties.

b) Logical level information
   1) Logical devices representing application parts with their descriptive properties,
   2) End points of logical connections attached to logical devices representing the application part interface with their descriptive properties,
   3) Logical connections representing the information exchange between application parts with their descriptive properties.
   4) Process data units modelling the exchanged data packages.

c) Relations between logical and physical levels
   1) Mapping of end points to represent the implementation of logical connections by physical connections,
   2) Relations between variables exchanged and applied communication technology/protocol especially by mapping of variables to communication package structures.

d) Communication networks
   1) Unique association of devices and connections to networks,
   2) Unique hierarchical network structuring,
   3) Dependencies between networks,
   4) Dependencies between devices such as redundancy properties,
   5) Dependencies between connections such as redundancy properties.

5 UML model

5.1 Overview

The UML model uses class diagrams showing the static relationships between network entities. The entities are always described as types. The real implementation of the model depends on tool preferences. An aggregation demonstrates here a "has a" association or relationship between the classes and the aggregation is more specific than association [1].

More important in the UML class diagrams are the defined cardinalities. If the cardinality is not defined then it is to be read as 0...n.

Figure 17 shows the focal points of the modelled communication structure of a single communication network. The physical device is the central object of the further descriptions. In particular, the modelled structure is divided into the logical and physical topological views.

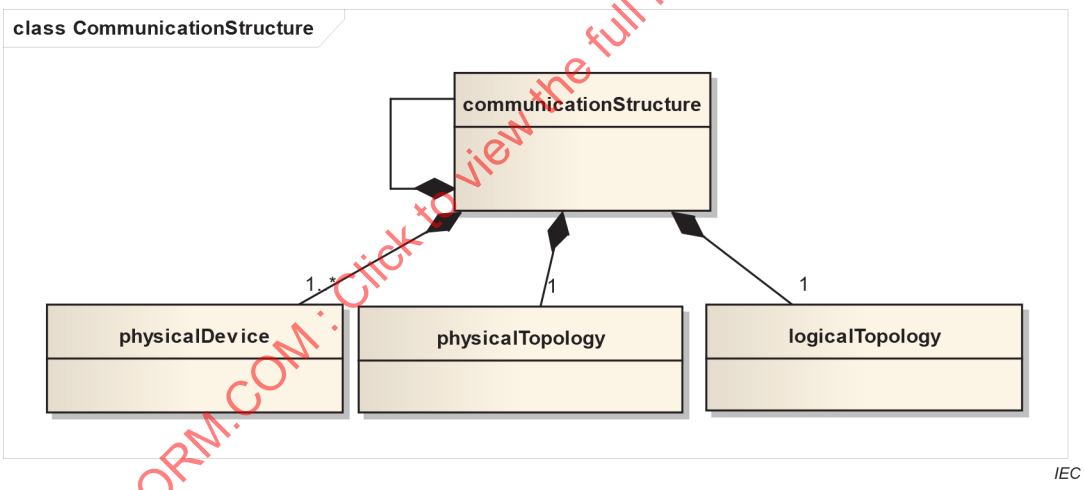


Figure 17 – Structure of communication network

The relationship of the communication structure to itself (reflexive association) allows the representation of nested communication networks or the representation of several interconnected network segments.

5.2 Logical topology

5.2.1 Aim of logical topology

The logical topology describes the logical view of the communication partners. It represents communication relations from the point of view of application programs. Figure 18 shows the three involved (abstract) items and their relationships:

- logicalTopology,
- logicalConnection,
- logicalEndPoint.

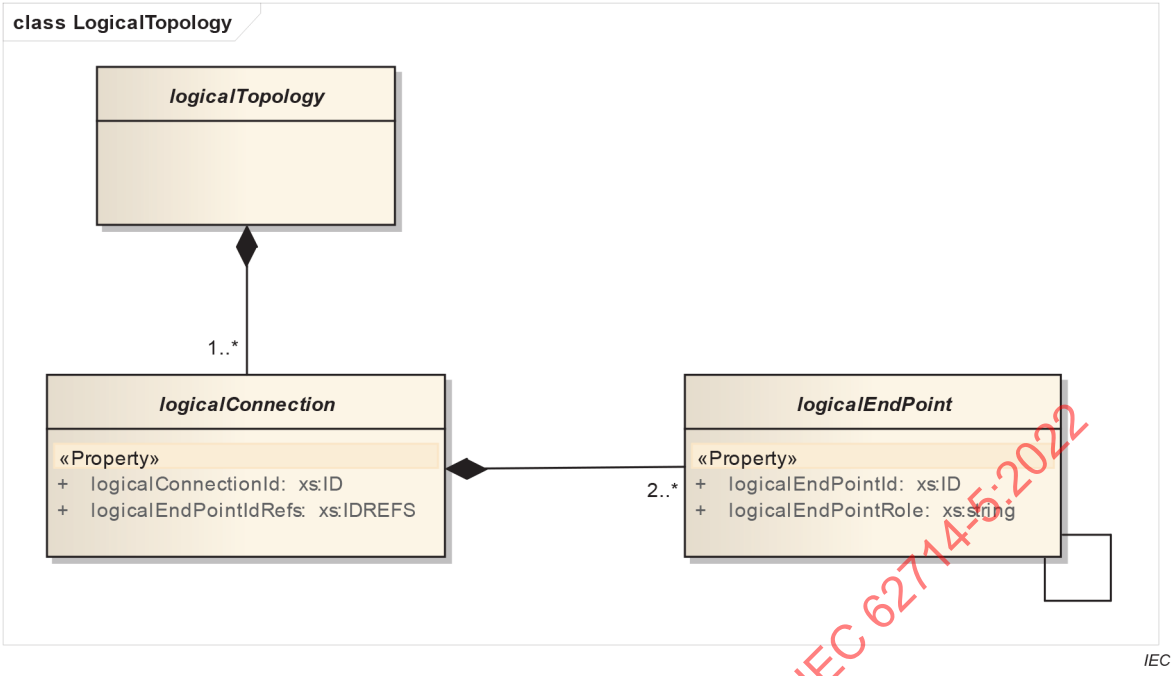


Figure 18 – View on logical topology

5.2.2 Item logicalTopology

The class logicalTopology allows the modelling of an automation or communication network on top of a physical network. It shall aggregate all communication relations between logical end points, which in turn are associated to physical devices.

A real derivation of logicalTopology may be the representation of a control network in a plant.

5.2.3 Item logicalConnection

The entity logicalConnection shall describe the communication relation between two logical end points, which are communication partners. A more generic term is ‘logical link’, indicating the relationship between at least two logicalEndPoints.

The property logicalConnectionId is a GUID within the communicationStructure and shall identify the logicalConnection.

The property logicalEndPointIdRefs is an array and shall contain all GUIDs of logicalEndPoints related to the logicalConnection. Thus, m:n relationships in a communication relation can be modelled.

A real derivation of logicalConnection may be a connection relation in a peer-to-peer-relationship.

5.2.4 Item logicalEndPoint

The entity logicalEndPoint shall represent a communication partner in such a relationship.

The property logicalEndPointId is a GUID within the communicationStructure and shall identify the logicalEndPoint.

The property logicalEndPointRole is of data type string and shall be used to specify the logical role of the logicalEndPointId in the communication relation. Conceivable values could be pairs such as "source" – "destination", "source" – "sink", "publisher" – "subscriber" or "master" – "slave", etc.

Derivations of such a communication end point may be an IP port or a peer.

5.3 Physical topology

5.3.1 Aim of physical topology

The physical topology describes the physical view on the communication partners. It shall represent communication relations from the point of view of wiring or channel settings. Figure 19 shows the three involved (abstract) items and their relationships:

- physicalTopology,
- physicalConnection,
- physicalEndPoint.

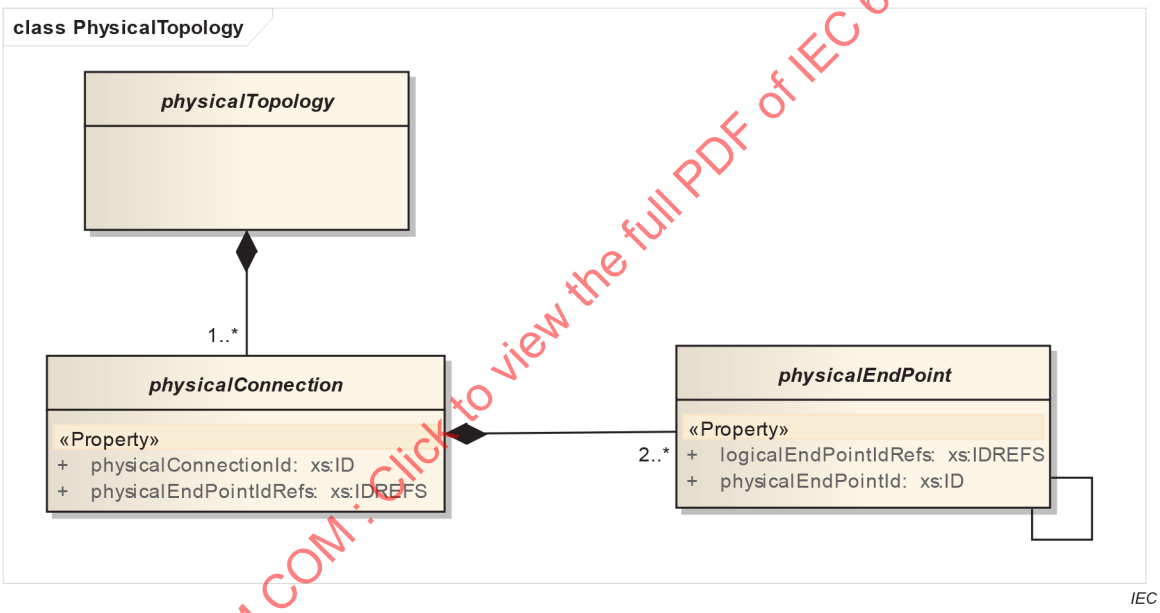


Figure 19 – View on physical topology

5.3.2 Item physicalTopology

The class physicalTopology is the entry point for modelling the physical topology of an automation network. It shall aggregate all communication relations between physical end points, associated with automation devices.

A real derivation of physicalTopology may be the representation of physical networks in a plant.

5.3.3 Item physicalConnection

The entity physicalConnection shall describe the communication relation between two physical end points. The physicalEndPoints may also be connected by means of a wireless network. In this case a link via the ether is established.

The property physicalConnectionId is a GUID within the communicationStructure and shall identify the physicalConnection.

The property physicalEndPointIdRefs is an array and shall contain all GUIDs of physicalEndPoints related to the physicalConnection. Thus, m:n connections in a communication network can be modelled.

A real derivation of physicalConnection may be a cable connecting sockets in a peer-to-peer-relationship.

5.3.4 Item physicalEndPoint

The entity physicalEndPoint shall represent a real communication endpoint in such a relationship.

The property physicalEndPointId is a GUID within the communicationStructure and shall identify the physicalEndPoint.

The property logicalEndPointIdRefs is an array and shall contain GUIDs of logicalEndPoints related to this physicalEndPoint.

Derivation of such an end point may be an Ethernet port.

5.4 Device

5.4.1 General

The model of the device is the most complex in the UML model. Figure 20 shows the first part and Figure 21 the second part of it.

5.4.2 Item physicalDevice

Central component is the physicalDevice. The attributes allow the unique identification of the instances and sub-instances of physicalDevice. Furthermore, there is an aggregation relationship to itself. By means of this relation, modular devices can be modelled. The aggregation of the information item summarises all important product data descriptions, identifiers, documentation, etc. as a central object, which may also be used by instances of a logicalDevice. A physicalDevice or its sub modules aggregate the logicalDevices. Furthermore, the physical end points, eg. ports, are assigned to the physicalDevice by means of the physicalEndPointList. The same pattern is used for the physical channels. In the case that the physicalDevice is a controller, it may contain several deviceResources.

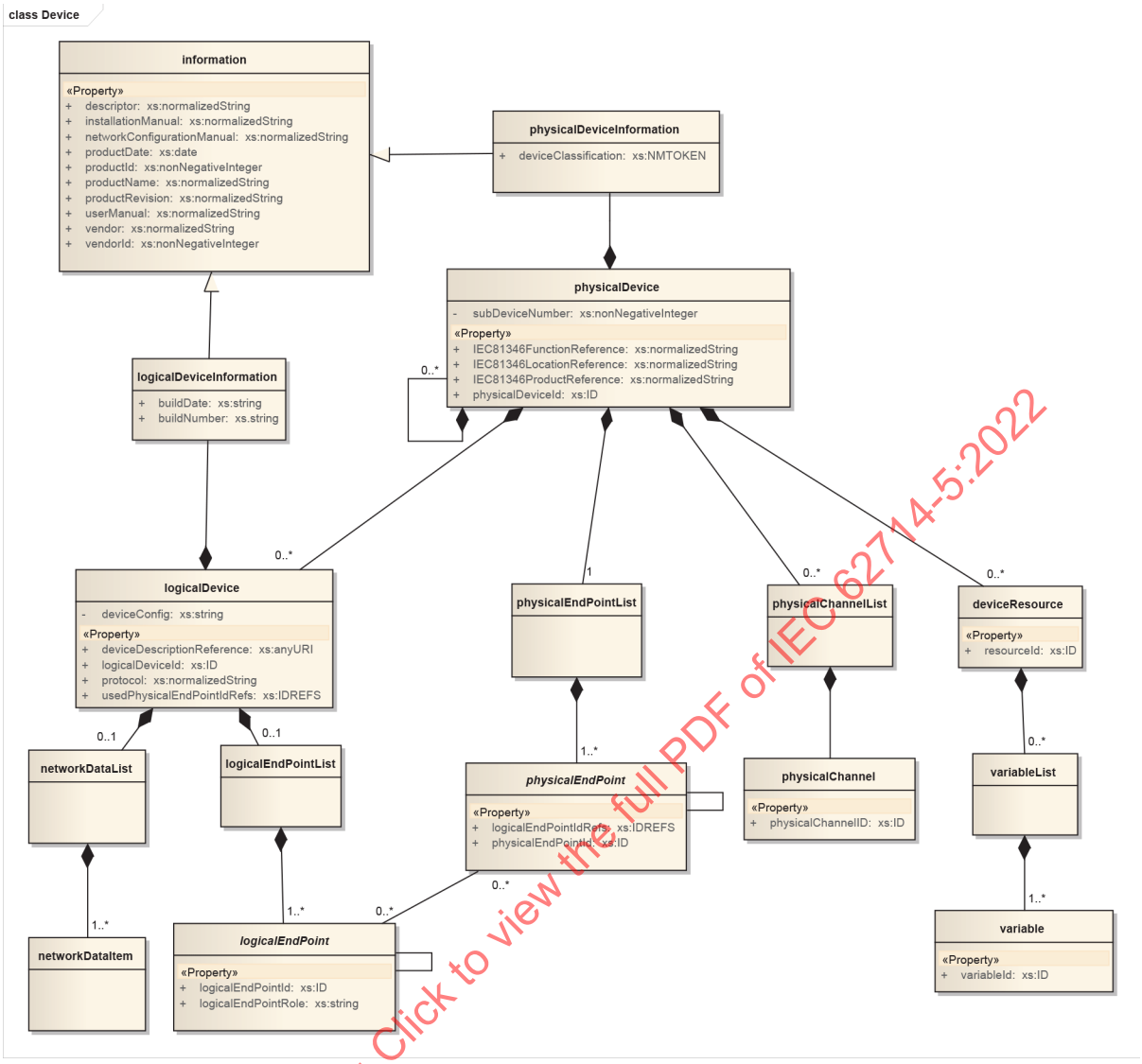


Figure 20 – Part 1 of the device model

NOTE The item logicalEndPoint is the transition point to Figure 21 and is repeated there.

The property physicalDeviceId is a GUID within the communicationStructure and shall identify the physicalDevice. If it is a submodule inside a device, the property subDeviceNumber shall define the manufacturer specific order of the submodule in the device.

The property IEC81346FunctionReference shall describe the functional aspect of the physicalDevice according to the rules of IEC 81346.

The property IEC81346LocationReference shall describe the location of the physicalDevice in the plant according to the rules of IEC 81346.

The property IEC81346ProductReference shall describe the product information of the physicalDevice according to the rules of IEC 81346.

5.4.3 Item Information

The item Information is a placeholder for any kind of information associated with the physical and logical device type. It shall be used to describe the type information of a device instance in detail. The proposed properties may be enhanced on demand.

5.4.4 Item physicalDeviceInformation

The item physicalDeviceInformation expands the generic placeholder Information and shall represent device information related to the physicalDevice.

5.4.5 Item logicalDeviceInformation

The item logicalDeviceInformation expands the generic placeholder Information and shall represent device information related to the logicalDevice.

5.4.6 Item logicalDevice

The item logicalDevice shall represent non-physical aspects of a real device. Often more complex field devices contain several logicalDevices with their own address information and specific tasks.

The logicalDevice shall aggregate the networkDataItems via the networkDataList, used for configuration tasks. Also the logicalEndPoints shall be aggregated via the logicalEndpointlist. In addition to the representation of the logical topology in 4.2, the logicalEndPoint also shall aggregate the protocol data units, exchanged via the logical connection.

The property deviceConfig shall describe information used during start-up of the logicalDevice. They may be exchanged at runtime between different communication partners, depending on the communication system.

The property deviceDescriptionReference shall reference to an external device description file, if available.

The property logicalDeviceId is a GUID inside the automation system and shall identify the logicalDevice.

The property protocol shall describe the used communication protocol, e.g. Ethernet based industrial bus, fieldbus etc.

The property usedPhysicalEndPointIdRefs shall represent a list of GUID with associated physicalEndPoints. This means that these end points will be used by the logicalDevice.

5.4.7 Item networkDataList

The item networkDataList shall represent the networkDataItems associated to the logicalDevice.

5.4.8 Item networkDataItem

A networkDataItem shall represent one specific configuration item that belongs to the logicalDevice. Typical networkDataItems can be MAC address, ip address, subnet mask, station name etc.

5.4.9 Item logicalEndPointList

This item is a pattern to enumerate all available logicalEndPoints.

5.4.10 Item physicalEndPointList

This item is a pattern to enumerate all available physicalEndPoints.

5.4.11 Item physicalChannelList

The item physicalChannelList shall represent a placeholder for elements of physicalChannel.

5.4.12 Item physicalChannel

The item physicalChannel shall represent a connecting point to a signal, e.g. an analogue or binary in- or output. The data type is not explicitly modelled here. The physicalChannel is listed via physicalChannelList.

The property physicalChannelId is a GUID within the communicationStructure and shall identify the channel or signal.

5.4.13 Item deviceResource

The item deviceResource shall represent a runtime environment (e.g. a CPU or virtual machine with specific software) inside a device allowing executing programs or parts of programs with its man-machine interface or sensor and actuator interface functions. The item is a representation of the term RESOURCE used in IEC 61131-3.

5.4.14 Item variableList

This item is a pattern to enumerate all available variables inside a deviceResource.

5.4.15 Item variable

The item variable represents a variable e.g. of a controller program or in another type of controller. The data type is not explicitly modelled here. The item variable shall represent a VARIABLE inside a programmable component, e.g. a controller according to the definition of IEC 61131-3. The item variable is already defined in IEC 62714-4 as an interface and references to a variable element of PLCopen XML (IEC 61131-10) by means of the GUID.

The property variableId is a GUID within the communicationStructure and shall identify the variable.

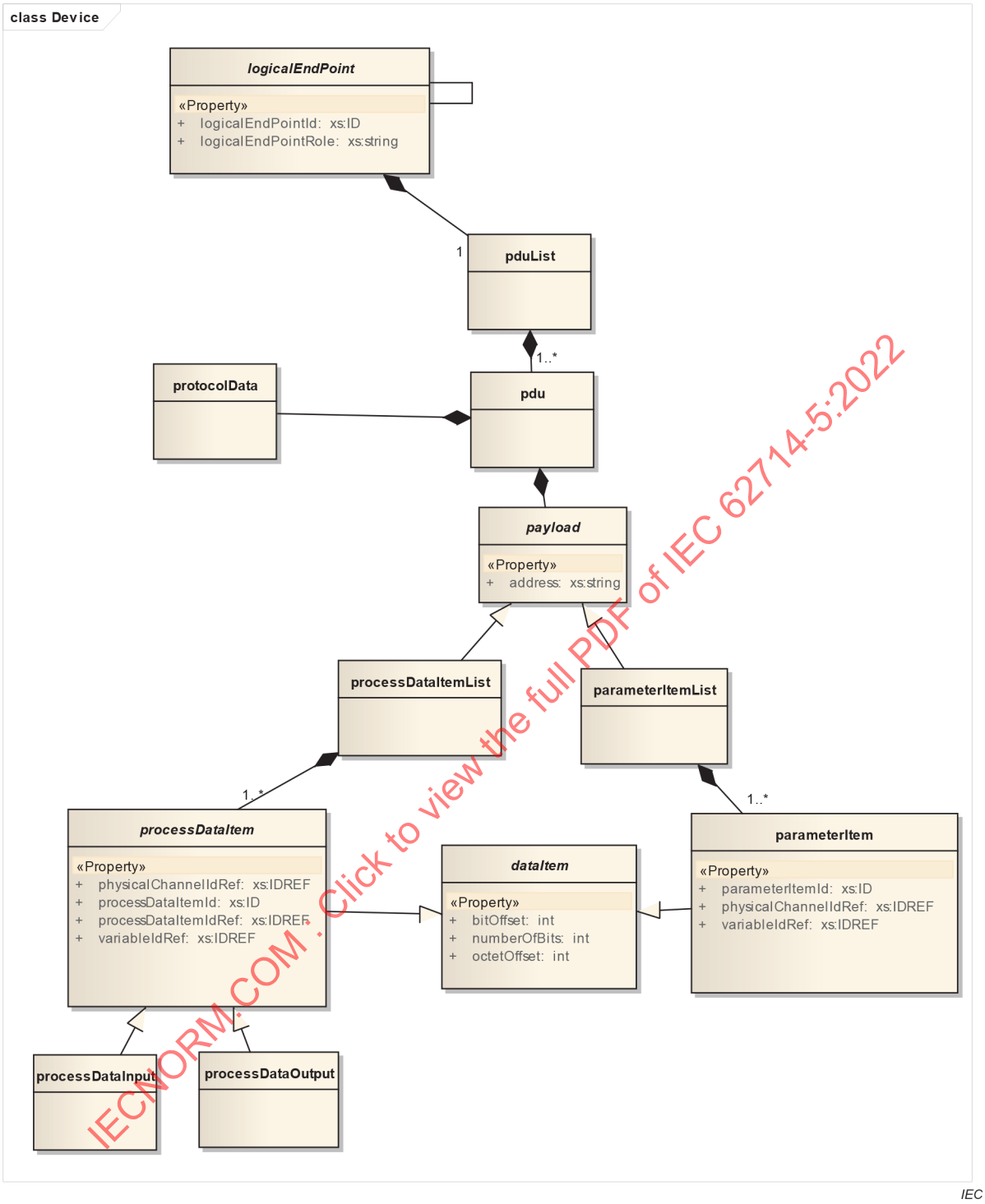


Figure 21 – Part 2 of the device model

NOTE The item logicalEndPoint is the transition point from Figure 20 and is repeated here.

5.4.16 Item pduList

The item pduList shall represent the Protocol Data Units exchanged via the logicalEndPoint.

5.4.17 Item pdu

The item pdu shall represent one specific Protocol Data Unit related to this logicalConnection. It shall aggregate the protocolData and the payload. The payload can be specialised into processDataItemList or parameterItemList.

5.4.18 Item protocolData

The item protocolData shall be used to describe the protocol overhead, which is normally not modelled by the processDataItem or parameterItem. For real communication protocols, specialisations of these items can be defined.

5.4.19 Item payload

The item payload shall represent the processDataItem and parameterItem as a generalized abstract item. The optional property address may be used for address or identification purposes of a dataItem.

5.4.20 Item processDataItemList

This item is a pattern to enumerate all processDataItems.

5.4.21 Item parameterItemList

This item is a pattern to enumerate all parameterItems. In addition, a pdu of this type shall have a communication protocol specific address.

5.4.22 Item dataItem

The item dataItem shall represent a data object communicated via the logicalConnection.

The property octetOffset shall define the octet offset of the data object in the PDU payload. The counting shall start with the first octet of the payload starting with index ‘0’ for first octet to send/receive.

The property bitOffset shall define the bit offset of the data object inside an addressed octet of the PDU payload.

The properties octetOffset and bitOffset may be used alternatively. If both properties are used, they shall be added in order to calculate the bit position.

The property numberOfBits shall define the length of the data object in bits.

5.4.23 Item processDataItem

The item processDataItem shall represent a process data object communicated via the logicalConnection. Such data objects are often transmitted by means of cyclic service primitives of the communication system.

The property processDataItemId is a GUID inside the automation system and shall identify the processDataItem.

The property variableIdRef shall be the reference of the processDataItem inside the process data image associated with a variable of the controller or controller program. The reference is set, if the physicalDevice is a controller.

The property processDataItemIdRef shall be the reference of the processDataItem inside the process data image associated with a variable of the device program. The reference is set, if the physicalDevice is not a controller.

The property physicalChannelIdRef shall be the reference of the processDataItem to a physicalChannel of a device.

5.4.24 Item processDataInput

The item processDataInput is a specialisation of processDataItem and shall represent the data flow of the process data as an input from the controlled process.

5.4.25 Item processDataOutput

The item processDataOutput is a specialisation of processDataItem and shall represent the data flow of the process data as an output to the controlled process.

5.4.26 Item parameterItem

The item parameterItem shall represent a parameter object communicated via the logicalConnection. Such data objects are often transmitted by means of acyclic service primitives of the communication system.

The property parameterItemId is a GUID inside the automation system and shall identify the parameterItem.

The property variableItemIdRef shall be the reference to a variable of the controller program. The reference is set, if the physicalDevice is a controller.

The property physicalChannelIdRef shall be the reference of the processDataItem to a physicalChannel of a device.

6 Representation within AutomationML

6.1 Overview of mapping

6.1.1 Introduction of mapping

The modelling method for communication system modelling within AutomationML is based on the use of dedicated <RoleClass>es and <InterfaceClass>es as well as a special application process. This method will take up the distinction between application networks and physical networks and its interconnection as described in Clauses 4 and 5.

6.1.2 General mapping rules

For representation of the different entities named in Clauses 4 and 5, the following mapping rules as shown in Table 1 will be applied. It is distinguished in what manner the semantic of an object is represented and a related entity is modelled.

Table 1 – Mapping rules

Information entity of UML model	Semantic modelling	Entity modelling
logicalTopology	RoleClass LogicalNetwork	InternalElement
logicalConnection	RoleClass LogicalConnection	InternalElement
logicalEndpoint	InterfaceClass LogicalEndPoint	Interface
physicalTopology	RoleClass PhysicalNetwork	InternalElement
physicalConnection	RoleClass PhysicalConnection	InternalElement
physicalEndpoint	InterfaceClass PhysicalEndPoint	Interface
physicalDevice	RoleClass PhysicalDevice	InternalElement
variable	InterfaceClass VariableInterface as defined in IEC 62714-4	Interface
physicalChannel	InterfaceClass SignalInterface as defined in IEC 62714-1	Interface
logicalDevice	RoleClass LogicalDevice	InternalElement
pduList	Implicitly modelled by hosting InternalElement	
pdu	RoleClass CommunicationPackage	InternalElement
protocolData	Attribute or InterfaceClass DatagrammObject	Attribute or Interface
payload	InterfaceClass DatagrammObject	Interface
processDataItemList	Implicitly modelled by hosting InternalElement	
parameterItemList	Implicitly modelled by hosting InternalElement	
dataItem	InterfaceClass DatagrammObject	Interface
processDataItem	InterfaceClass DatagrammObject	Interface
parameterItem	InterfaceClass DatagrammObject	Interface

6.1.3 Basics

The basis of the modelling method is the definition of an AutomationML role class library and an AutomationML interface class library for communication system modelling and the derivation from them of relevant role classes and interface classes for the special application cases following the definition of role class and interface class application defined in IEC 62714-1.

The AutomationML communication role class library will contain roles dedicated to identifying internal elements as physical devices, physical connections and physical networks as well as internal elements such as logical devices, logical connections, and logical networks.

The AutomationML interface class library will contain interface classes for physical end points and logical end points.

Both communication role class library and the communication interface class library are depicted in Figure 22.

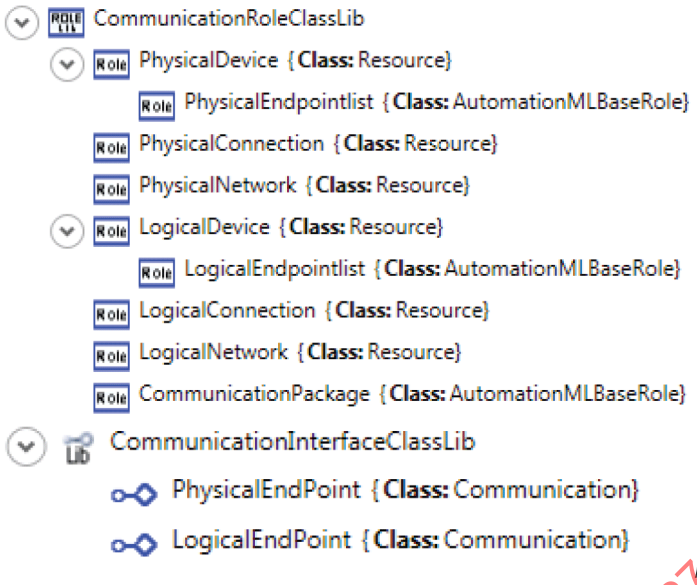


Figure 22 – Communication role class library and communication interface class library

Exploiting these basic roles and interface classes, special classes can be derived identifying devices and connections related to special applications and communication technologies. Role class libraries and interface class libraries for special purposes will be developed. An example is given in Figure 23.

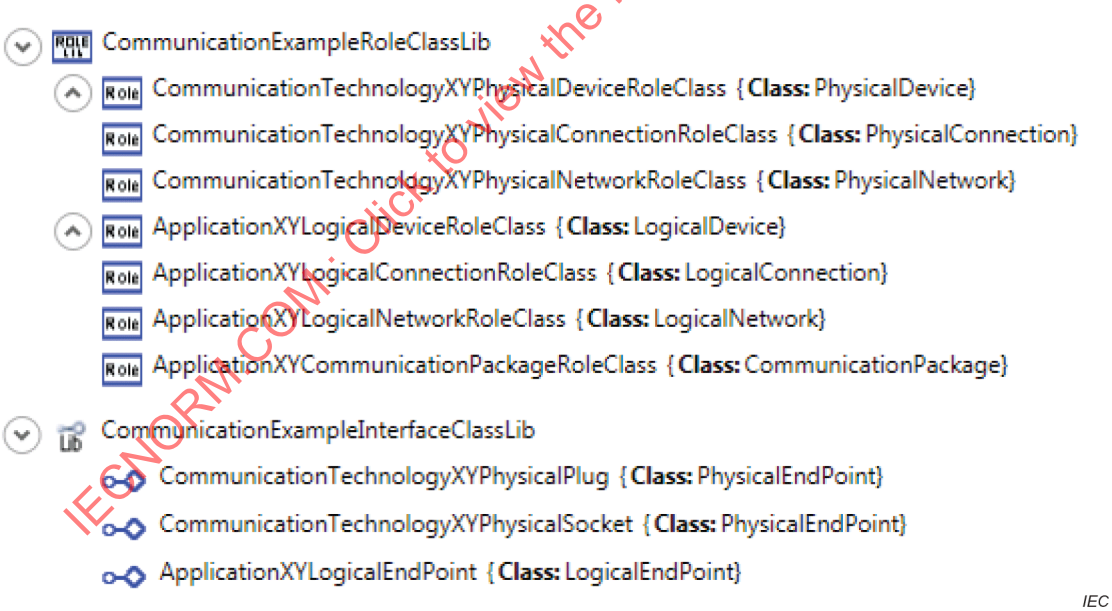


Figure 23 – Derived role class libraries and interface class libraries for a special example

6.1.4 Modelling of relations

In UML diagrams (Figure 20, Figure 21), relations are modelled between classes expressed by means of GUIDs and list of GUIDs. In AutomationML, such relations are explicitly modelled by means of CAEX InternalLinks, see Table 2.

Table 2 – Modelling of relations in AutomationML

Attributes in UML model	AutomationML modelling
logicalConnection::logicalEndPointIdRefs	One or multiple CAEX InternalLinks between two logicalEndPoints
physicalConnection::physicalEndPointIdRefs	One or multiple CAEX InternalLink between two physicalEndPoints
physicalEndPoint::logicalEndPointIdRefs	One or multiple CAEX InternalLinks between one physicalEndPoint and one logicalEndPoint
logicalDevice::usedPhysicalEndPointIdRefs	One or multiple CAEX InternalLinks between one logicalDevice and one physicalEndPoint
processDataItem::variableIdRef	One CAEX InternalLink between one processDataItem and one PLCvariable
processDataItem::physicalChannelIdRef	One CAEX InternalLink between one processDataItem and one physicalChannel
processDataItem::processDataItemIdRef	One CAEX InternalLink between one processDataItem and one CAEX ExternalInterface related to the data item
parameterItem::physicalChannelIdRef	One CAEX InternalLink between one processDataItem and one physicalChannel
parameterItem::variableIdRef	One CAEX InternalLink between one parameterItem and one PLCvariable

6.1.5 Application process

The defined application case related role and interface class libraries are now applied for definition of usually applied physical devices and connections as well as logical devices and connections by defining appropriate <SystemUnitClass>es within related <SystemUnitClassLib>s. For the purpose of unique identification of the semantics of the different defined system unit classes, the defined role classes are used and referenced. In the application example, four system unit class libraries are defined for physical devices, logical devices, physical connections, and logical connections (see Figure 24).

Each physical device will be equipped with as many physical end point objects as physical ports are available which are integrated in an Endpointlist <InternalElement>. Each logical device will be equipped with as many logical end point objects as logical application access points are provided. They are integrated in an Endpointlist <InternalElement>.

Each physical connection object will be equipped with as many physical end point objects as the connection can connect on physical devices. Usually, in case of physical wiring, these are two end point objects. Each logical connection object will be equipped with as many logical end point objects as the connection can connect on logical devices. In case of master slave communication, these are two end point objects, in case of multicast communication, these can be more than two end point objects.

For representation of special properties of the different system unit classes, appropriate attributes should be applied.

Based on the developed system unit class libraries, the communication system of interest can now be modelled. Therefore, all necessary physical and logical devices are instantiated as <InternalElement>s in an appropriate <InstanceHierarchy>. Especially the hierarchical structure of the modelled system shall be reflected/preserved. This is especially valid for the integration of logical devices within physical devices as shown in Figure 25 for the logical device IODevice1 within the physical device AXLBKPN.

After defining all devices, the relevant attributes of the devices are to be completed and filled with values.

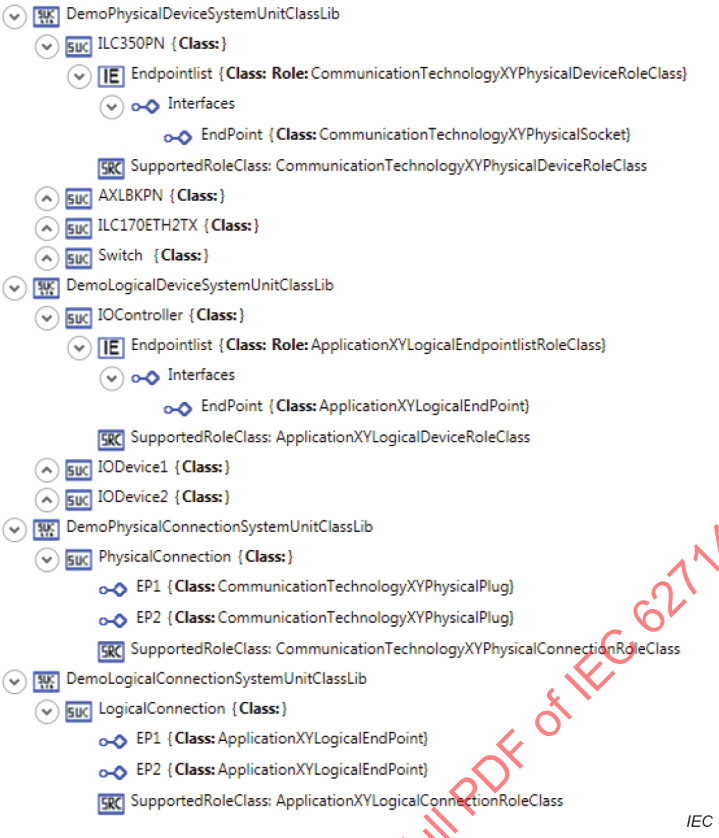


Figure 24 – SystemUnitClassLib examples for communication system modelling

When the devices have been completed, they can be connected by connections. Therefore, in the <InstanceHierarchy> of the network, two <InternalElement>s are instantiated containing the roles <PhysicalNetwork> and <LogicalNetwork>. They will act as containers for all physical and logical connection objects. Within the internal element with role <PhysicalNetwork> one <InternalElement> with role physical connection is instantiated for each physical connection out of the appropriate system unit class library. It is completed by appropriate attributes and its values. Additionally, within the internal element with role <LogicalNetwork> one <InternalElement> with role logical connection is instantiated for each logical connection out of the appropriate system unit class library. It is completed by appropriate attributes and its values.

When all necessary devices and connections are instantiated, they will be connected by exploiting <InternalLink>s. Therefore, for each logical device and each logical connection, which are interconnected, the related logical end points will be interrelated by an internal link object. Also for each physical device and each physical connection, which are interconnected, the related physical end points will be interrelated by an internal link object. To also map the logical end points to physical end points implementing the related connections, internal link objects are used for connecting physical end points with logical end points.

For the example, the resulting structure is depicted in Figure 25.

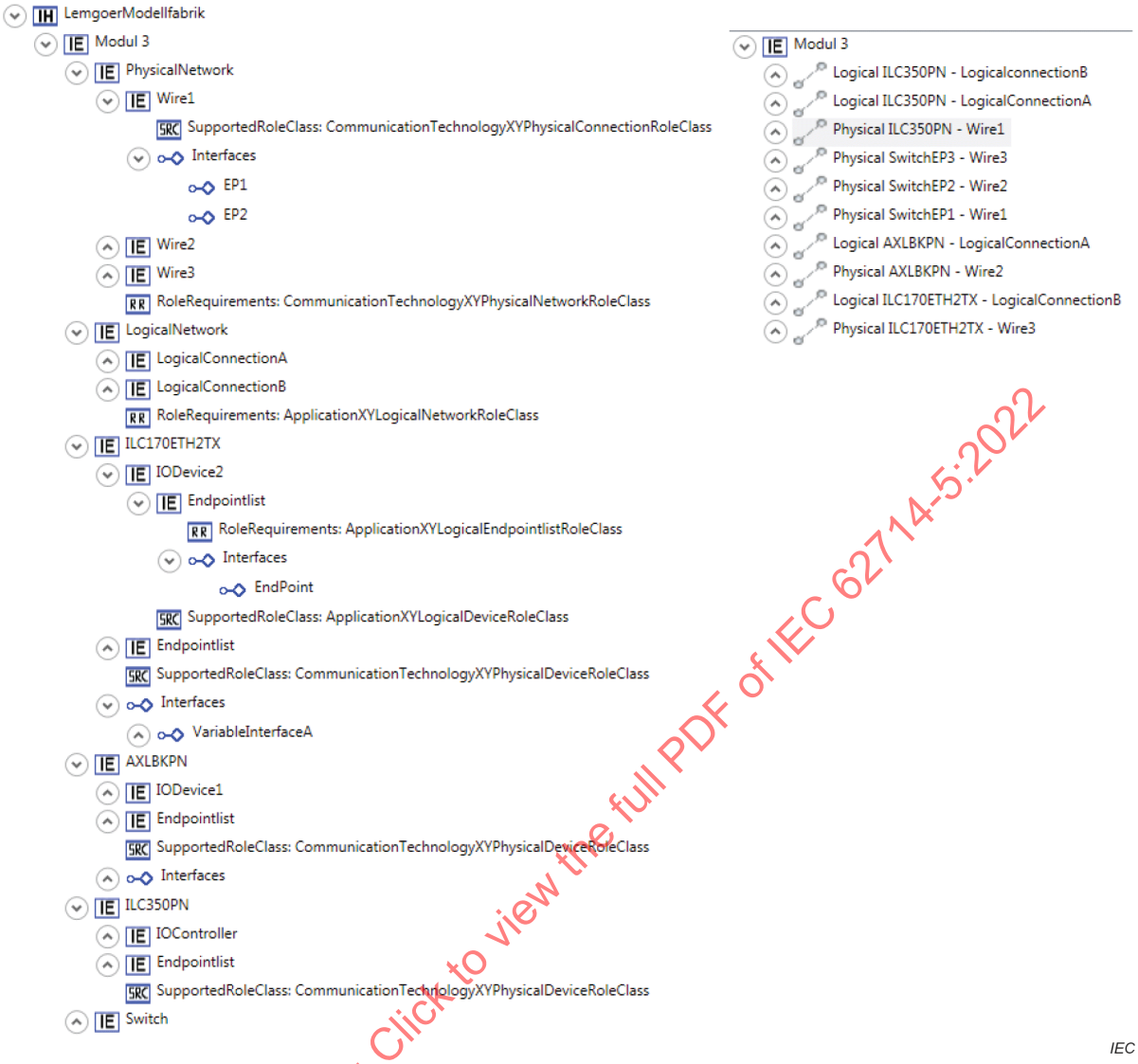


Figure 25 – Final network model example

NOTE Depending on the application case, parts of the modelling process named above can be omitted. When modelling only physical networks, the definition of logical network entity role classes, interface classes, and system unit classes can be neglected. In addition, a sequential system modelling starting with either the physical or the logical network and adding the other one later on is possible.

6.2 Basic communication role class library

6.2.1 General

This subclause defines a base library of essential standard role classes required for the modelling of core communication concepts (Figure 26, Figure 27 and Figure 28). All roles are technology independent but the basis for technology dependent libraries. Details of each role class are given in 6.2.2 to 6.2.9.

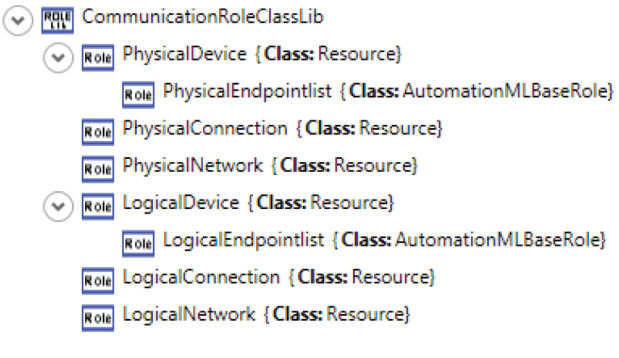


Figure 26 – Basic communication role class library

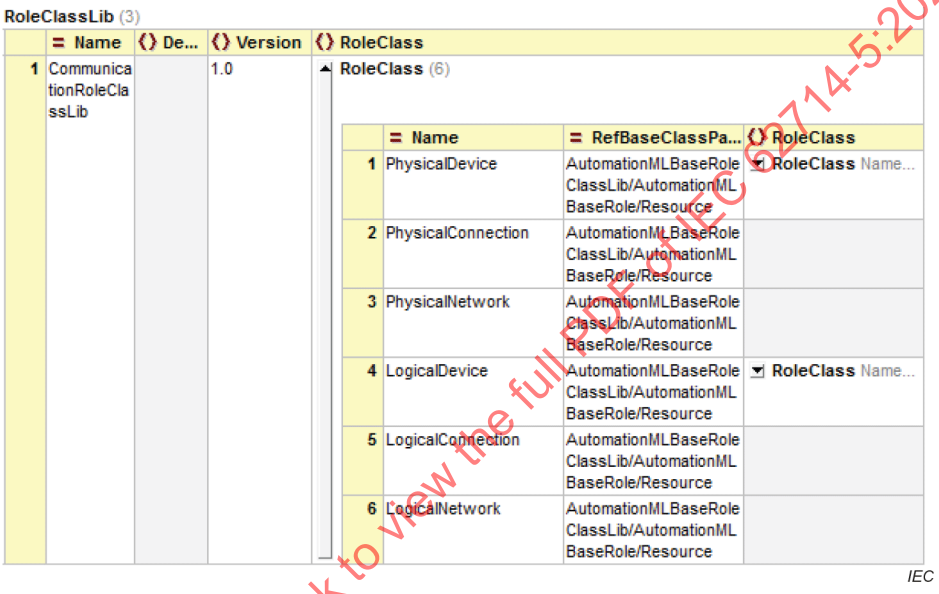


Figure 27 – CommunicationRoleClassLib

```
<RoleClassLib Name="CommunicationRoleClassLib">
  <Version>1.0</Version>
  <RoleClass Name="PhysicalDevice" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource">
    <RoleClass Name="PhysicalEndpointlist" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole" />
  </RoleClass>
  <RoleClass Name="PhysicalConnection" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="PhysicalNetwork" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="LogicalDevice" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource">
    <RoleClass Name="LogicalEndpointlist" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole" />
  </RoleClass>
  <RoleClass Name="LogicalConnection" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="LogicalNetwork" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
</RoleClassLib>
```

IEC

Figure 28 – XML text of the communication role class library

6.2.2 RoleClass PhysicalDevice

Table 3 defines the role class "PhysicalDevice".

Table 3 – RoleClass PhysicalDevice

Class name	PhysicalDevice	
Description	The role class "PhysicalDevice" is a role type for objects that represent physical devices which are part of the communication system. All objects describing physical devices in a communication system shall directly or indirectly reference this role. Examples for physical devices are IO boards, controllers or Profibus coupler.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.2.3 RoleClass PhysicalEndpointlist

Table 4 defines the role class "PhysicalEndpointlist".

Table 4 – RoleClass PhysicalEndpointlist

Class name	PhysicalEndpoinlist	
Description	The role class "PhysicalEndpointlist" models a hierarchy element collecting physical interfaces of a physical device. Since a physical device may contain one or more physical communication interfaces (e.g. physical sockets), a physical port list shall be a child of a physical device and all physical communication interfaces shall be children of the port list object. This allows modelling of physical devices with multiple communication interfaces, e.g. Ethernet based industrial bus, fieldbus etc.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

6.2.4 RoleClass PhysicalConnection

Table 5 defines the role class "PhysicalConnection".

Table 5 – RoleClass PhysicalConnection

Class name	PhysicalConnection	
Description	The role class "PhysicalConnection" is an abstract role type describing a physical connection between the physical interfaces of physical devices, e.g. a cable or a wireless communication channel.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.2.5 RoleClass PhysicalNetwork

Table 6 defines the role class "PhysicalNetwork".

Table 6 – RoleClass PhysicalNetwork

Class name	PhysicalNetwork	
Description	The role class "PhysicalNetwork" is a role type describing the network of physical connections in a communication system, e.g. a list of wires and their interfaces.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.2.6 RoleClass LogicalDevice

Table 7 defines the role class "LogicalDevice".

Table 7 – RoleClass LogicalDevice

Class name	LogicalDevice	
Description	The role class "LogicalDevice" is a role type for objects that represent logical devices. A physical device may contain multiple logical devices which are connected to existing physical interfaces of the physical device.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.2.7 RoleClass LogicalEndpointlist

Table 8 defines the role class "LogicalEndpointlist".

Table 8 – RoleClass LogicalEndpointlist

Class name	LogicalEndpointlist	
Description	The role class "LogicalEndpointlist" models a hierarchy element collecting logical interfaces of a logical device. Since a logical device may contain one or more logical communication interfaces, a logical port list shall be a child of a logical device and all logical communication interfaces shall be children of the port list object. This allows modelling of logical devices with multiple communication interfaces.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

6.2.8 RoleClass LogicalConnection

Table 9 defines the role class "LogicalConnection".

Table 9 – RoleClass LogicalConnection

Class name	LogicalConnection	
Description	The role class "LogicalConnection" is an abstract role type describing a logical connection between logical interfaces of logical devices.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.2.9 RoleClass LogicalNetwork

Table 10 defines the role class "LogicalNetwork".

Table 10 – RoleClass LogicalNetwork

Class name	LogicalNetwork	
Description	The role class "LogicalNetwork" is a role type describing the network of logical connections in a communication system.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

6.3 Basic communication interface class library

6.3.1 General

This subclause defines a base library of essential standard role classes required for the modelling of core communication concepts (Figure 29, Figure 30, Figure 31). All roles are technology independent but the basis for technology dependent libraries. Details of each interface class are given in 6.3.2 and 6.3.3.

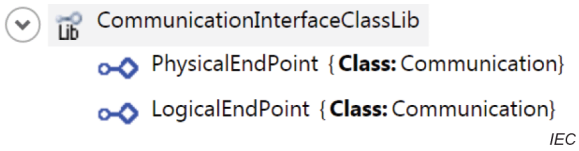


Figure 29 – Basic communication interface class library

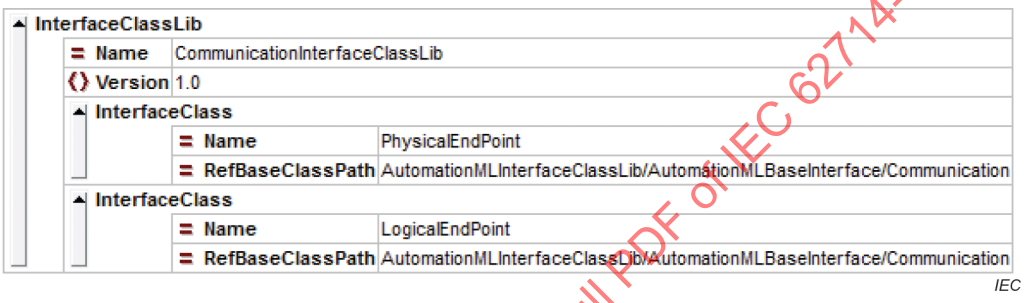


Figure 30 – CommunicationInterfaceClassLib

```
<InterfaceClassLib Name="CommunicationInterfaceClassLib">
        <Version>1.0</Version>
        <InterfaceClass Name="PhysicalEndPoint"
        RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication"/>
        <InterfaceClass Name="LogicalEndPoint"

        RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication"/>
</InterfaceClassLib>
```

IEC

Figure 31 – XML text of the communication interface class library

6.3.2 InterfaceClass PhysicalEndPoint

Table 11 defines the interface class "PhysicalEndPoint".

Table 11 – InterfaceClass PhysicalEndPoint

Class name	PhysicalEndPoint	
Description	The interface class "PhysicalEndPoint" is an interface type to be used for modelling physical end points, e.g. physical sockets, of a physical object. Instances of this class shall be subelements of Endpointlists of physicalDevices.	
Parent Class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributes	None	

6.3.3 InterfaceClass LogicalEndPoint

Table 12 defines the interface class "LogicalEndPoint".

Table 12 – InterfaceClass LogicalEndPoint

Class name	LogicalEndPoint	
Description	The interface class "LogicalEndPoint" is an interface type to be used for modelling logical end points. Instances of this class shall be subelements of Endpointlists of logicalDevices	
Parent Class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributes	None	

6.4 Steps to model technology specific libraries

6.4.1 General

The described basic role and interface classes are still independent from any technology. In order to model communication systems based on concrete technologies such as Ethernet based industrial bus, fieldbus, etc, corresponding technology specific classes shall be derived. This subclause is about the definition of technology specific libraries in 3 steps. This stepwise approach is intended for a progressive development of the communication system model.

Logical and physical networks need not be developed at the same time.

The development of corresponding classes is explained by means of an abstract technology CommunicationTechnologyXY and an application ApplicationXY. Libraries for concrete technologies are out of the scope of this document.

6.4.2 Step 1: Development of technology specific role classes

For the first step, a technology specific role class library shall be created whereby each of the specific communication role classes is derived from the corresponding basic class. In this example, the resulting library is named CommunicationExampleRoleClassLib (see Figure 32).

Figure 32 – Derivation of a technology specific role class library out of the base role class library

EXAMPLE ProfinetPhysicalDeviceRoleClass, ProfinetPhysicalConnectionRoleClass, IOLinkPhysicalDeviceRoleClass, IOLinkPhysicalConnectionRoleClass.

In practice, for each concrete communication technology such as Ethernet based industrial bus, fieldbus, etc, an individual library should be developed based on the physical network related roles. In addition, for each concrete application such as device configuration service, drive control, line station control, etc., a separate library can be developed based on the logical network related roles.

The naming of role classes and their parameters shall be done in accordance with the appropriate technology standards.

6.4.3 Step 2: Development of technology specific interface classes

In step two, a technology specific interface class library shall be created whereby each of the specific communication interface classes is derived from the corresponding basic class. Since most communications are based on plugs and sockets, in this example, the class PhysicalEndPoint leads to two classes. For this example, the resulting library is named CommunicationExampleInterfaceClassLib (see Figure 33).

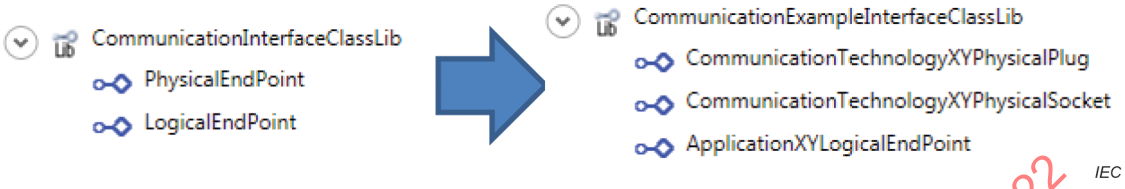


Figure 33 – Derivation of a technology specific role class library out of the base role class library

EXAMPLE ProfinetPhysicalPlug, ProfinetPhysicalSocket IOLinkPlug, IOLinkSocket.

The naming of interface classes and its parameters shall be done in accordance with the appropriate technology standards.

6.4.4 Step 3: Development of system unit class libraries

Based on the technology specific role and interface class libraries, all necessary physical devices and physical connections shall be modelled as <SystemUnitClass> in a <SystemUnitClassLib> which references the corresponding role class. Their interfaces refer to the corresponding interface classes.

The following example models three physical devices with one physical interface each, a switch with 6 interfaces, and three logic devices. The resulting modelled connection classes are depicted in Figure 34.

The naming of system unit classes and their parameters shall be done in accordance with the appropriate technology standards.

The modelling of specific attributes of the individual physical or logical devices and connections or their interfaces is done by means of CAEX attributes which are attached to corresponding <InternalElement>s, <SystemUnitClass>es or <InterfaceClass>es as shown in Figure 34.

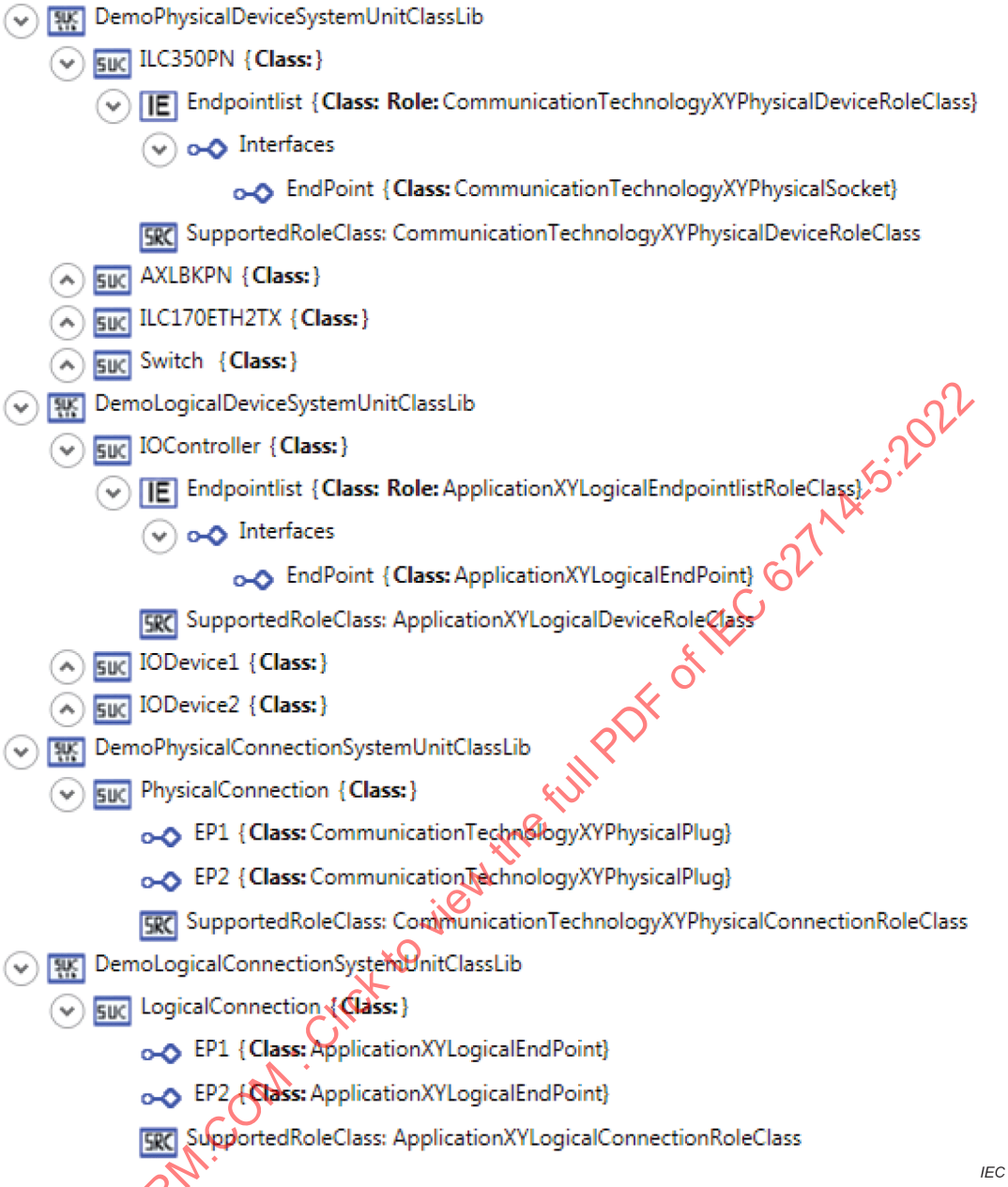


Figure 34 – Technology specific <SystemUnitClassLib>s

6.4.5 Step 4: Modelling the network

After modelling the technology specific classes, Step 4 pursues the modelling of the actual logical and physical devices. For this, all required physical and logical devices shall be instantiated as new <InternalElement>s in the <InstanceHierarchy> in their required hierarchical order. All relevant attributes get relevant values in order to describe individual properties of the devices.

EXAMPLE: ProfinetNetwork,consisting of LogicalNetwork, PhysicalNetwork, Controller_A, Device_B, Device_C, see Figure 7.

6.4.6 Step 5: Modelling the connections

The final step serves for the interconnection of the devices. The physical and logical networks each shall be modelled as an individual <InternalElement> in the <InstanceHierarchy> referring to the role class <PhysicalNetwork> or <LogicalNetwork> respectively. They serve as a container for the physical or logical connections, which shall be created by instantiating the corresponding <SystemUnitClass>es and by assigning their individual parameters. The relations between the devices and connections are finally modelled as CAEX <InternalLink>s interlinking related interfaces. Figure 35 shows an example of a communication network.

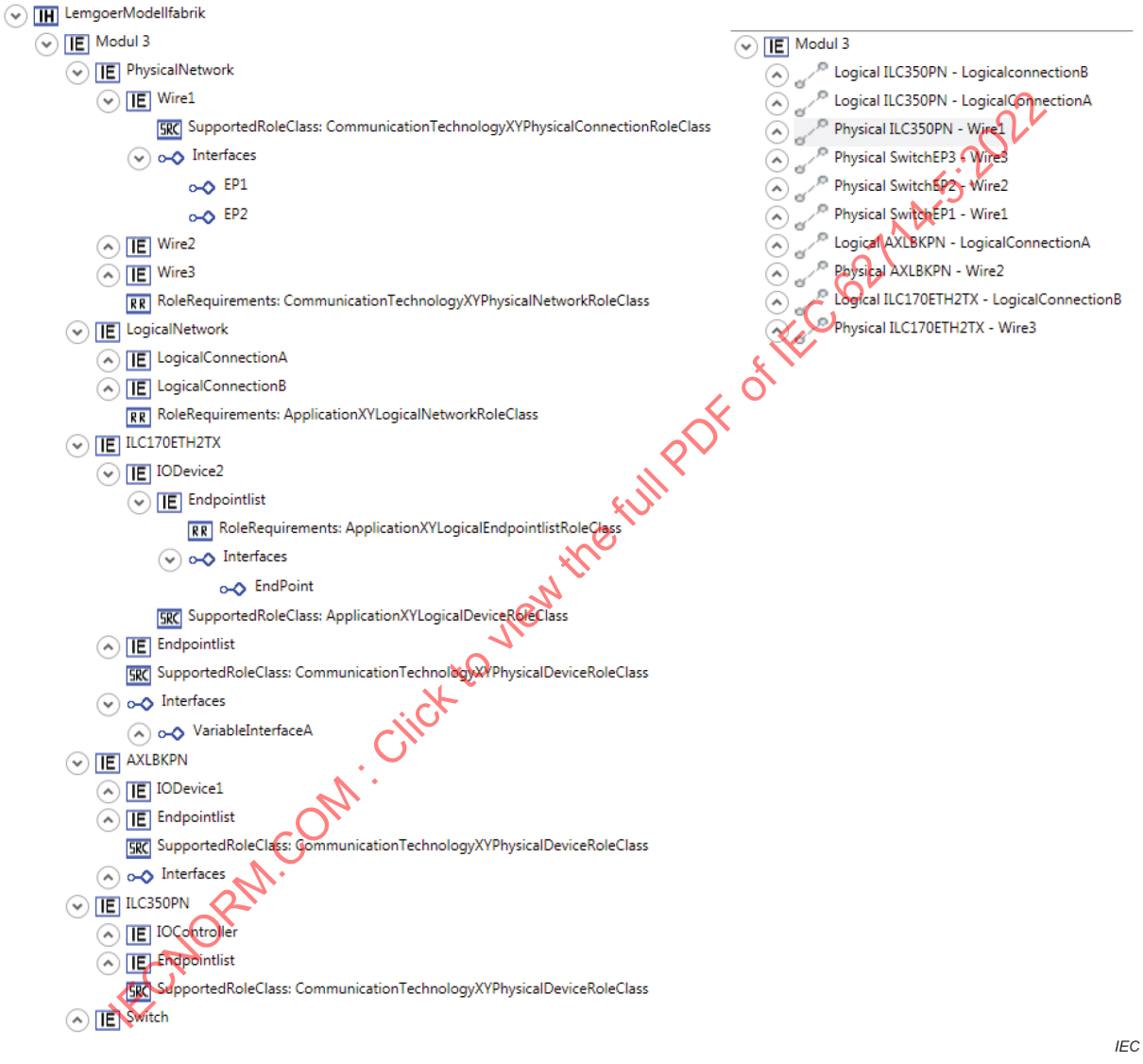


Figure 35 – Technology specific communication network

6.5 PDU modelling

6.5.1 General

Following 4.3.5 and 5.4, logical devices will exchange communication data packages (PDU) via their logical interfaces.

This subclause defines an extension of the communication libraries by essential standard role classes and essential interface classes as well as the necessary modelling steps for modelling PDUs within a communication system model, which are required for modelling communication data packages.

6.5.2 RoleClass CommunicationPackage

The basic communication role class library is extended by an additional role CommunicationPackage as described in Table 13 and represented in Figure 36, Figure 37 and Figure 38.

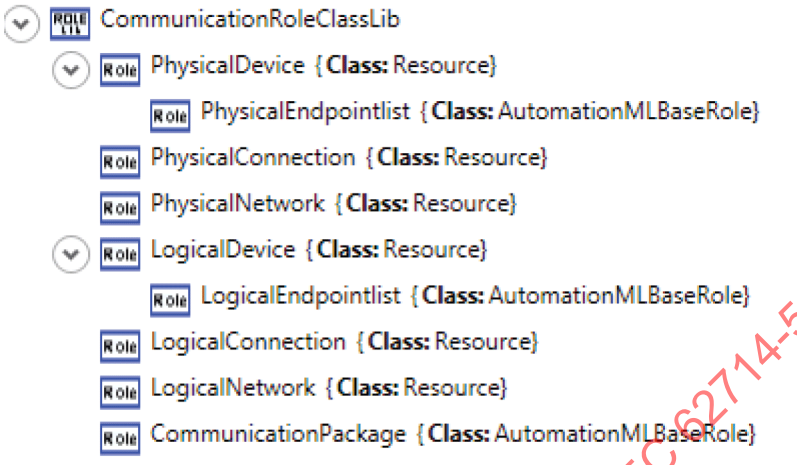


Figure 36 – Extended communication role class library

	Name	D	Version	RoleClass
1	CommunicationRoleClassLib		1.0	RoleClass (7)

	Name	RefBaseClassPath	RoleClass
1	PhysicalDevice	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	RoleClass: Name = PhysicalEndpointlist; RefBaseClassPa... = AutomationMLBaseRoleClassLib/AutomationMLBaseRole
2	PhysicalConnection	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
3	PhysicalNetwork	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
4	LogicalDevice	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	RoleClass: Name = LogicalEndpointlist; RefBaseClassPa... = AutomationMLBaseRoleClassLib/AutomationMLBaseRole
5	LogicalConnection	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
6	LogicalNetwork	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
7	CommunicationPackage	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	

IEC

Figure 37 – Extended CommunicationRoleClassLib

```
<RoleClassLib Name="CommunicationRoleClassLib">
  <Version>1.0</Version>
  <RoleClass Name="PhysicalDevice" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource">
    <RoleClass Name="PhysicalEndpointlist" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole" />
  </RoleClass>
  <RoleClass Name="PhysicalConnection" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="PhysicalNetwork" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="LogicalDevice" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource">
    <RoleClass Name="LogicalEndpointlist" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole" />
  </RoleClass>
  <RoleClass Name="LogicalConnection" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="LogicalNetwork" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource" />
  <RoleClass Name="CommunicationPackage" RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole" />
</RoleClassLib>
```

IEC

Figure 38 – XML text of the extended communication role class library

Table 13 defines the role class "CommunicationPackage".

Table 13 – RoleClass CommunicationPackage

Class name	CommunicationPackage	
Description	The role class "CommunicationPackage" is a role type for objects that represent communication packages exchanged via logical connections between logical devices within a communication system. All objects describing communication packages in a communication system shall directly or indirectly reference this role. Examples for communication packages are Ethernet datagrams or Fieldbus packages.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

6.5.3 InterfaceClass DatagrammObject

The basic communication interface class library is extended by an additional interface class DatagrammObject as described in Table 14 and represented in Figure 39, Figure 40 and Figure 41.

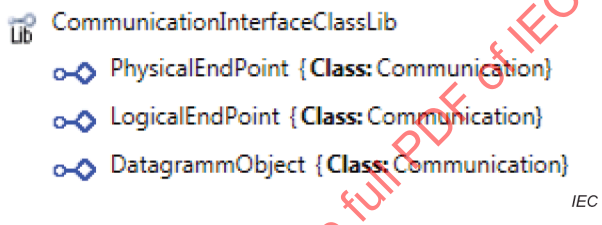


Figure 39 – Extended communication interface class library

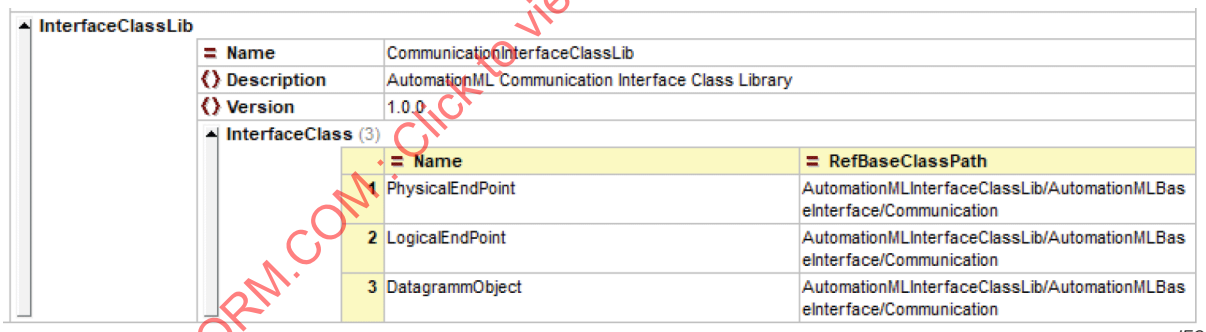


Figure 40 – Extended CommunicationInterfaceClassLib

```
<InterfaceClassLib Name="CommunicationInterfaceClassLib">
  <Version>1.0</Version>
  <InterfaceClass Name="PhysicalEndPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication" />
  <InterfaceClass Name="LogicalEndPoint" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication" />
  <InterfaceClass Name="DatagrammObject" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication" />
</InterfaceClassLib>
```

IEC

Figure 41 – XML text of the extended communication role class library

Table 14 defines the interface class "DatagrammObject".

Table 14 – InterfaceClass DatagrammObject

Class name	DatagrammObject	
Description	The interface class "DatagrammObject" is an interface type to be used for modelling parts of a communication package exchanged via a logical connection between logical devices. It represents a consistent information set exchanged which corresponds to the information set represented by a PLCopenXMLInterface as defined in IEC 612714-4 of the AutomationML specification.	
Parent Class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributes	None	

6.5.4 Steps to model technology specific libraries

6.5.4.1 General

As described in 6.4, the extended role and interface classes are still independent from any technology. In order to model communication systems based on concrete technologies such as Ethernet based industrial bus, fieldbus, etc, corresponding technology specific classes have to be derived. This subclause is about the extension of the definition of technology specific libraries given in 6.5 to model PDUs.

The development of corresponding classes is explained by means of an abstract technology named CommunicationTechnologyXY and an abstract application ApplicationXY. Libraries for concrete technologies are out of the scope of this document.

6.5.4.2 Step 1: Development of technology specific role classes

The first step given in 6.4.2 is extended for PDU modelling by deriving an additional technology dependent role class for the CommunicationPackage as given in Figure 42. Thereby, an ApplicationXYCommunicationPackageRoleClass is derived from CommunicationPackage role class.

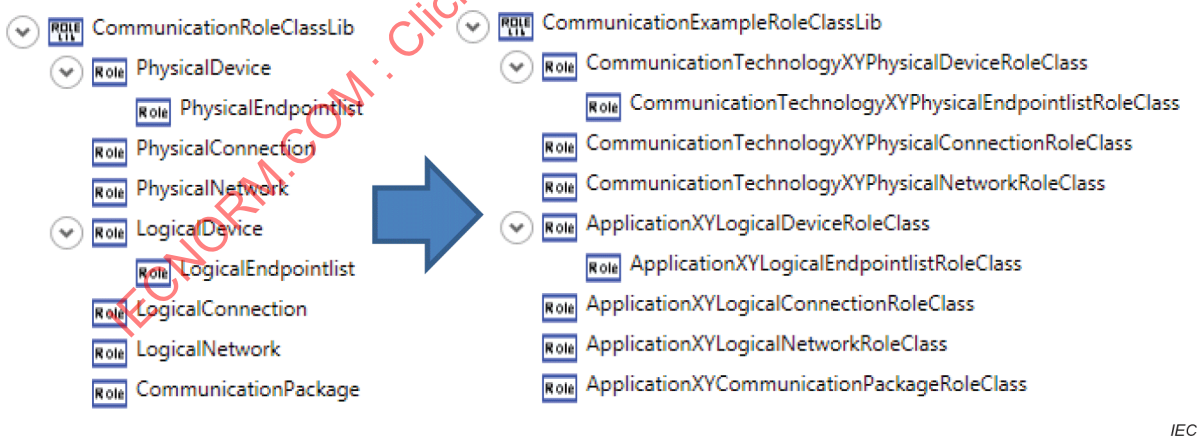


Figure 42 – Derivation of a technology specific role class library out of the extended role class library

6.5.4.3 Step 2: Development of technology specific interface classes

The second step given in 6.4.3 is extended for PDU modelling by deriving an additional technology dependent interface class for the data object transmitted with a communication package as given in Figure 43. Thereby, an ApplicationXYDatagrammObject interface class is derived from DatagrammObject interface class.

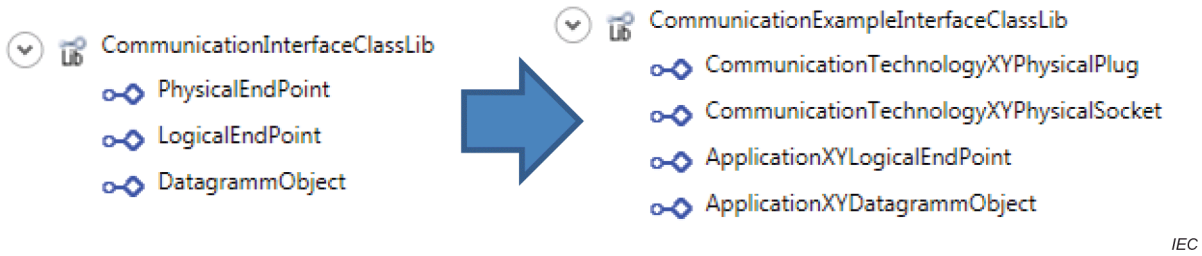


Figure 43 – Derivation of a technology specific interface class library out of the extended interface class library

6.5.4.4 Step 3: Development of system unit class libraries

The third step given in 6.4.4 is extended for PDU modelling by deriving an additional technology dependent system unit class for the communication package as given in Figure 44. Thereby, an ApplicationXYDataPackage system unit class is defined.

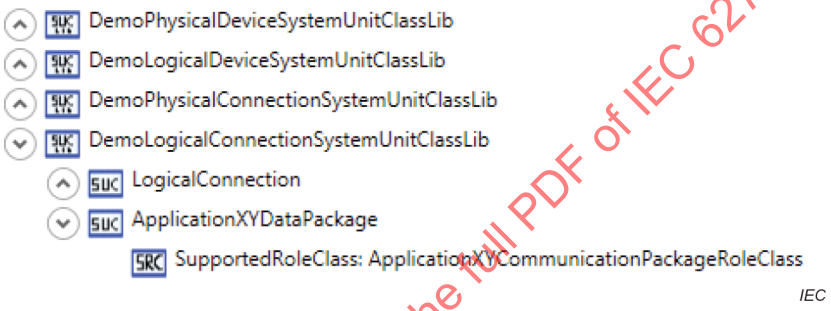


Figure 44 – Technology specific extended <SystemUnitClassLib>s

6.5.4.5 Step 4 and 5 – Modelling of communication network

Steps 4 and 5 of the communication system modelling remain unchanged.

6.5.4.6 Step 6: Modelling the communication packages

An additional Step 6 is added to model the communication packages.

Within this step, for each communication package transmitted via a logical connection within the related Internal Element with the role ApplicationXYLogicalConnectionRoleClass, an Internal Element is derived from the system unit class ApplicationXYDataPackage. Within this Internal Element, for each information object transmitted and to be modelled, an Interface is derived from the interface class ApplicationXYDatagrammObject.

To model PDU, related properties as well as datagram object properties attributes for the corresponding internal elements and interfaces are used.

The interface of the type ApplicationXYDatagrammObject is connected by two internal links with the interfaces of PLCopenXMLinterface type or SignalInterface type representing the exchanged information on sender and receiver side.

This structure is depicted in Figure 45.

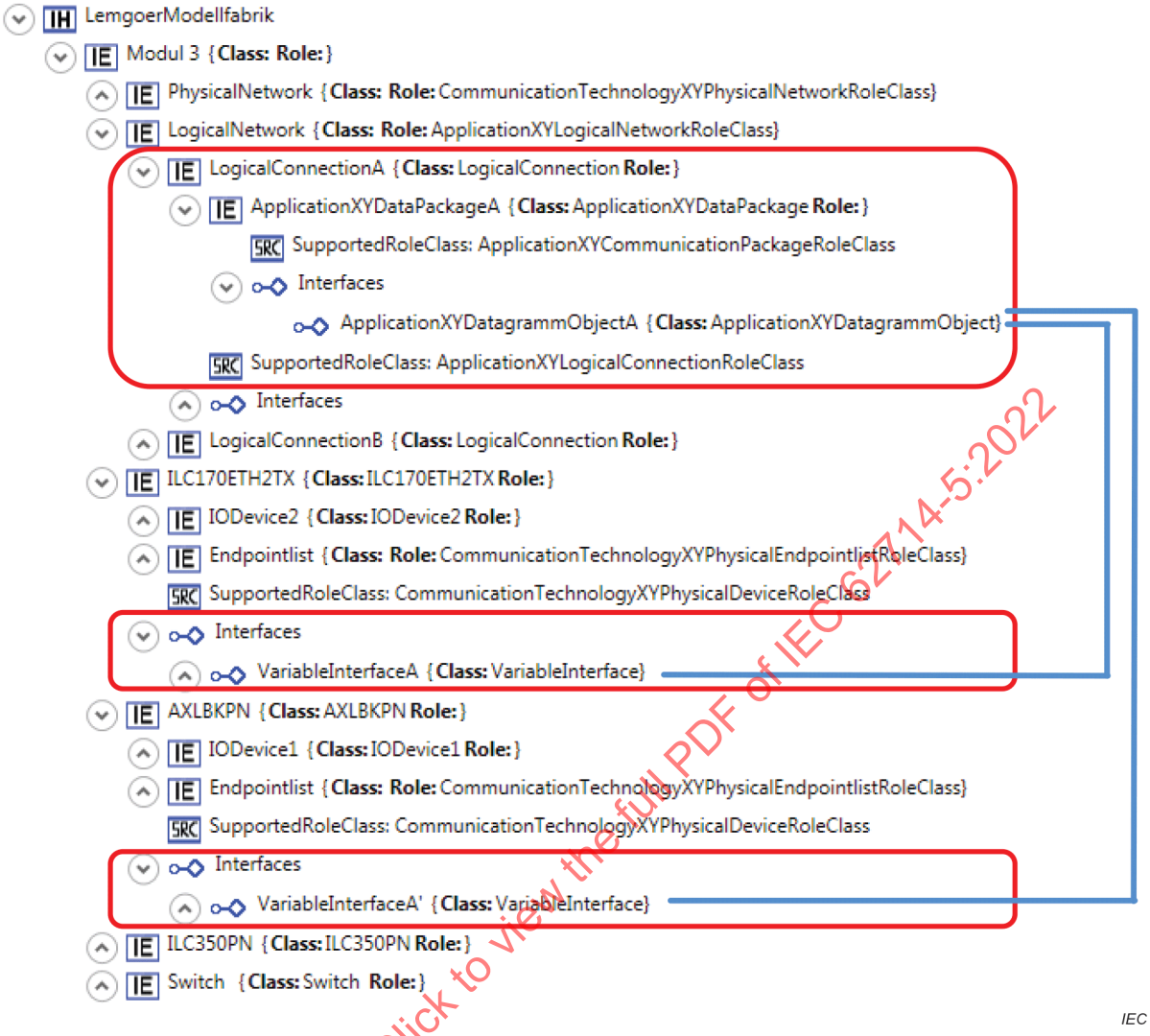


Figure 45 – Technology specific communication network with communication package models

6.6 References to attributes

As given in 6.4.4 and 6.5.4.6, communication device, interface, connection, PDU, and datagram object related attributes are modelled by means of CAEX attributes. In Table 15, examples of possible attributes related to communication systems are named and described.

Table 15 – Communication related attributes

Attribute type name	Description	DataType, value range	Use limitations	Example
Address	The parameter *address* specifies any address information (e.g. IEC address, IP address, port number, OPCTag, MAC address) of an entity (e.g. a signal).	String		10.7.8.12
AdressOffSet	The attribute *address offset* specifies which bit in the IO memory map of the assigned IO unit the IO signal is mapped to. The offset of the first IO is 0.	UINT	Not for logical and physical devices	8.5
BitCount	The attribute *bit count* specifies the length of the data item in bits. The minimum value is 1 for a Boolean signal. The value can be derived from the *IEC data type*, if available.	UINT >0		8
BitOffset	This attribute specifies which bit of a certain byte in the IO memory map of the assigned IO unit the IO signal is mapped to. The offset of the first bit in the first byte IO is 0.	Derived from AdressOffSet	Not for logical and physical devices	5
BitOrder	The attribute *bit order* specifies the meaning of the first bit (most or least significant). Allowed values are: "MSBFirst", "LSBFirst"	Enumeration "MSBFirst", "LSBFirst"		MSBFirst
OctetOffset	This attribute specifies the octet number in the IO memory map of the assigned IO unit the IO signal is mapped to. The octet offset of the first IO is 0. The octets are counted from the lowest octet.	Derived from AdressOffSet	Not for logical and physical devices	8
Direction	The attribute *direction* specifies the direction of the signal, either from the controller perspective (hard-wired IOs) or from the master device perspective (fieldbus connections).	Enumeration with values "In", "Out", "InOut"	Not for logical and physical devices	IN
EncodingType	Byteorder of encoding of data in the AML document and inside the PDUs at runtime	Enumeration BigEndian, LittleEndian		BigEndian
IECDataType	The parameter *IEC data type* specifies the IEC 61131-3 data type of the signal.	Enumeration out of IEC61131-3: BOOL, BYTE, WORD, DWORD, LWORD, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, REAL, LREAL, TIME, LTIME, DATE, LDATE		WORD

Attribute type name	Description	DataType, value range	Use limitations	Example
InitialValue	The parameter *initial value* specifies the IO signal value to be used at start.	Bytearray according to the IECDataType		0
LogicalUnit	The attribute *logical unit* specifies the engineering unit of the device that is connected to the analog input or output (e.g. "mm" for a distance sensor or "%" for a valve).	String	Only for logical end points	°C
MaxLogicalValue	This attribute specifies the logical value that will correspond to the *maximum physical value*. The logical values offer a way to access the IO signals by using logical quantities rather than physical. The attribute is only applicable for analog signals.	String (see NOTE)	Only for logical end points	30
MaxPhysicalValue	This attribute specifies the physical value that will correspond to the *maximum bit value*. The physical value directly corresponds to the value of the IO signal that this system attribute corresponds to, i.e. the amount of voltage or current given by a sensor. The attribute *maximum physical value* is only applicable for analog signals.	String (see NOTE)	Only for logical and physical end points and physical connections	20
MinLogicalValue	Minimum logical value of a measured physical value	String (see NOTE)	Only for logical end points	10
MinPhysicalValue	Minimum physical value of a measured physical value	String (see NOTE)	Only for logical and physical end points and physical connections	4
PhysicalUnit	The attribute physical unit specifies the unit of the IO signal. e.g. "mA", "V" The attribute physical unit is only applicable for analog signals.	String	Only for logical and physical end points and physical connections	mA
NOTE The format of the string content and the value range correspond to the data type specified in the attribute IECDataType.				

6.7 Usage of metadata

Metadata are modelled according to IEC 62424:2016, comprising information about the data source, version and revision information. The CAEX document providing the present communication related libraries shall have the following metadata regarding SourceDocumentInformation. Figure 46 defines the field "SourceDocumentInformation".

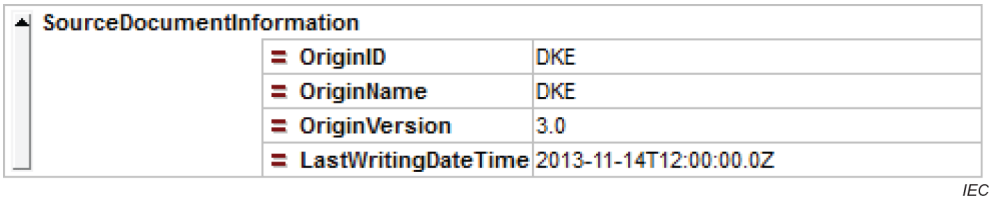


Figure 46 − Field SourceDocumentInformation according to communication related libraries

Version and revision related information shall be modelled according to IEC 62424:2016.

Bibliography

[1] Object Management Group: UML Class Diagram, http://www.uml.org/, 2012

[2] ZIMMERMANN, H., OSI Reference Model − The ISO Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications, Volume 28, Issue 4, pp. 425-432

SOMMAIRE

AVANT-PROPOS 60
1 Domaine d'application 62
2 Références normatives 62
3 Termes, définitions, termes abrégés et acronymes 63
3.1 Termes et définitions 63
3.2 Termes abrégés et acronymes 63
4 Cas d'utilisation et structures de réseau 64
4.1 Généralités 64
4.2 Cas d'utilisation 64
4.2.1 Activités d'ingénierie 64
4.2.2 Transfert sans perte d'informations sur l'instance du dispositif de communication 65
4.2.3 Transfert sans perte d'informations sur les systèmes de communication 69
4.3 Délimitation de la plage de modélisation 71
4.3.1 Domaine d'application de la plage de modélisation 71
4.3.2 Structures d'interaction et cycles de vie 71
4.3.3 Objets du réseau 72
4.3.4 Topologies de réseau 74
4.3.5 Contenu de la communication 79
4.4 Exigences de modélisation dérivées 79
5 Modèle de UML 80
5.1 Vue d'ensemble 80
5.2 Topologie logique 81
5.2.1 Objectif de la topologie logique 81
5.2.2 Élément logicalTopology 81
5.2.3 Élément logicalConnection 81
5.2.4 Élément logicalEndPoint 82
5.3 Topologie physique 82
5.3.1 Objectif de la topologie physique 82
5.3.2 Élément physicalTopology 82
5.3.3 Élément physicalConnection 83
5.3.4 Élément physicalEndPoint 83
5.4 Dispositif 83
5.4.1 Généralités 83
5.4.2 Élément physicalDevice 83
5.4.3 Élément Information 84
5.4.4 Élément physicalDeviceInformation 85
5.4.5 Élément logicalDeviceInformation 85
5.4.6 Élément logicalDevice 85
5.4.7 Élément networkDataList 85
5.4.8 Élément networkDataItem 85
5.4.9 Élément logicalEndPointList 85
5.4.10 Élément physicalEndPointList 85
5.4.11 Élément physicalChannelList 86
5.4.12 Élément physicalChannel 86
5.4.13 Élément deviceResource 86
5.4.14 Élément variableList 86

5.4.15 Élément variable 86
5.4.16 Élément pduList 87
5.4.17 Élément pdu 88
5.4.18 Élément protocolData 88
5.4.19 Élément payload 88
5.4.20 Élément processDataItemList 88
5.4.21 Élément ItemItemList 88
5.4.22 Élément dataItem 88
5.4.23 Élément processDataItem 88
5.4.24 Élément processDataInput 89
5.4.25 Élément processDataOutput 89
5.4.26 Élément parameterItem 89
6 Représentation dans AutomationML 89
6.1 Aperçu de la mise en correspondance 89
6.1.1 Introduction de la mise en correspondance 89
6.1.2 Règles générales de mise en correspondance 89
6.1.3 Principes de base 90
6.1.4 Modélisation des relations 91
6.1.5 Processus d'application 92
6.2 Bibliothèque de classes de rôles de communication de base 94
6.2.1 Généralités 94
6.2.2 RoleClass PhysicalDevice 95
6.2.3 RoleClass PhysicalEndpointlist 96
6.2.4 RoleClass PhysicalConnection 96
6.2.5 RoleClass PhysicalNetwork 96
6.2.6 RoleClass LogicalDevice 97
6.2.7 RoleClass LogicalEndpointlist 97
6.2.8 RoleClass LogicalConnection 97
6.2.9 RoleClass LogicalNetwork 97
6.3 Bibliothèque de classes d'interface de communication de base 98
6.3.1 Généralités 98
6.3.2 InterfaceClass PhysicalEndPoint 98
6.3.3 InterfaceClass LogicalEndPoint 98
6.4 Étapes de la modélisation des bibliothèques spécifiques à une technologie 99
6.4.1 Généralités 99
6.4.2 Étape 1: Développement de classes de rôles spécifiques à la technologie 99
6.4.3 Étape 2: Développement de classes d'interface spécifiques à la technologie 100
6.4.4 Étape 3: Développement de bibliothèques de classes d'unités de système 100
6.4.5 Étape 4: Modélisation du réseau 101
6.4.6 Étape 5: Modélisation des connexions 102
6.5 Modélisation des PDU 102
6.5.1 Généralités 102
6.5.2 RoleClass CommunicationPackage 103
6.5.3 InterfaceClass DatagrammObject 104
6.5.4 Étapes de la modélisation des bibliothèques spécifiques à une technologie 105
6.6 Références aux attributs 108

6.7 Utilisation des métadonnées 111
Bibliographie 112

Figure 1 – Activités générales d'ingénierie dans lesquelles le système de communication est intégré 64
Figure 2 – Flux d'informations du cas d'utilisation 66
Figure 3 – Flux d'informations alternatif du cas d'utilisation 68
Figure 4 – Flux d'informations du cas d'utilisation 70
Figure 5 – Exemple d'une vue de niveau logique sur les systèmes de communication 73
Figure 6 – Exemple d'une vue de niveau physique sur les systèmes de communication 74
Figure 7 – Vues combinées sur les systèmes de communication 74
Figure 8 – Exemple de topologie en étoile 75
Figure 9 – Exemple de topologie en anneau 75
Figure 10 – Exemple de topologie linéaire 76
Figure 11 – Réseau simple avec câblage direct 76
Figure 12 – Réseau avec infrastructure active 77
Figure 13 – Réseaux connectés par des passerelles 77
Figure 14 – Réseaux structurés hiérarchiques 78
Figure 15 – Réseau couvrant plusieurs applications 78
Figure 16 – Stratégie générale de modélisation des PDU 79
Figure 17 – Structure d'un réseau de communication 80
Figure 18 – Vue de la topologie logique 81
Figure 19 – Vue de la topologie physique 82
Figure 20 – Partie 1 du modèle de dispositif 84
Figure 21 – Partie 2 du modèle de dispositif 87
Figure 22 – Bibliothèque de classes de rôles de communication et bibliothèque de classes d'interface de communication 91
Figure 23 – Bibliothèques de classes de rôles et bibliothèques de classes d'interface dérivées pour un exemple particulier 91
Figure 24 – Exemples de SystemUnitClassLib pour la modélisation des systèmes de communication 93
Figure 25 – Exemple de modèle de réseau final 94
Figure 26 – Bibliothèque de classes de rôles de communication de base 95
Figure 27 – CommunicationRoleClassLib 95
Figure 28 – Texte XML de la bibliothèque de classes de rôles de communication 95
Figure 29 – Bibliothèque de classes d'interface de communication de base 98
Figure 30 – CommunicationInterfaceClassLib 98
Figure 31 – Texte XML de la bibliothèque de classes d'interface de communication 98
Figure 32 – Dérivation d'une bibliothèque de classes de rôles spécifiques à une technologie à partir de la bibliothèque de classes de rôles de base 99
Figure 33 – Dérivation d'une bibliothèque de classes de rôles spécifiques à une technologie à partir de la bibliothèque de classes de rôles de base 100
Figure 34 – <SystemUnitClassLib>s spécifiques à la technologie 101
Figure 35 – Réseau de communication spécifique à une technologie 102
Figure 36 – Bibliothèque de classes de rôles de communication étendue 103

Figure 37 – CommunicationRoleClassLib étendue 103
Figure 38 – Texte XML de la bibliothèque de classes de rôles de communication étendue 104
Figure 39 – Bibliothèque de classes d'interface de communication étendue 104
Figure 40 – CommunicationInterfaceClassLib étendue 105
Figure 41 – Texte XML de la bibliothèque de classes de rôles de communication étendue 105
Figure 42 – Dérivation d'une bibliothèque de classes de rôles spécifiques à une technologie à partir de la bibliothèque de classes de rôles étendue 106
Figure 43 – Dérivation d'une bibliothèque de classes d'interface spécifique à une technologie à partir de la bibliothèque de classes d'interface étendue 106
Figure 44 – <SystemUnitClassLib>s étendues spécifiques à la technologie 107
Figure 45 – Réseau de communication spécifique à la technologie avec modèles de paquetages de communication 108
Figure 46 – Champ SourceDocumentInformation selon les bibliothèques liées à la communication 111

Tableau 1 – Règles de mise en correspondance 90
Tableau 2 – Modélisation des relations dans AutomationML 92
Tableau 3 – RoleClass PhysicalDevice 96
Tableau 4 – RoleClass PhysicalEndpointlist 96
Tableau 5 – RoleClass PhysicalConnection 96
Tableau 6 – RoleClass PhysicalNetwork 96
Tableau 7 – RoleClass LogicalDevice 97
Tableau 8 – RoleClass LogicalEndpointlist 97
Tableau 9 – RoleClass LogicalConnection 97
Tableau 10 – RoleClass LogicalNetwork 97
Tableau 11 – InterfaceClass PhysicalEndPoint 98
Tableau 12 – InterfaceClass LogicalEndPoint 99
Tableau 13 – RoleClass CommunicationPackage 104
Tableau 14 – InterfaceClass DatagrammObject 105
Tableau 15 – Attributs liés à la communication 109

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

FORMAT D'ÉCHANGE DE DONNÉES TECHNIQUES POUR UNE UTILISATION DANS L'INGÉNIERIE DES SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – AUTOMATION MARKUP LANGUAGE –

Partie 5: Communication

AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets.

L'IEC 62714-5 a été établie par le sous-comité 65E: Les dispositifs et leur intégration dans les systèmes de l'entreprise, du comité d'études 65 de l'IEC: Mesure, commande et automation dans les processus industriels. Il s'agit d'une Norme internationale.

Le texte de cette Norme internationale est issu des documents suivants:

Projet	Rapport de vote
65E/844/FDIS	65E/886/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à son approbation.

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Le présent document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/standardsdev/publications.

Une liste de toutes les parties de la série IEC 62714, publiées sous le titre général *Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language,* peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu du présent document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer ce document en utilisant une imprimante couleur.

FORMAT D'ÉCHANGE DE DONNÉES TECHNIQUES POUR UNE UTILISATION DANS L'INGÉNIERIE DES SYSTÈMES D'AUTOMATISATION INDUSTRIELLE – AUTOMATION MARKUP LANGUAGE –

Partie 5: Communication

1 Domaine d'application

Les processus d'ingénierie des systèmes techniques et de leurs systèmes d'automatisation intégrés sont exécutés avec une efficacité et une qualité croissantes, d'autant plus que la durée du projet a tendance à augmenter avec la complexité du système d'ingénierie. Pour résoudre ce problème, le processus d'ingénierie est plus souvent exécuté en exploitant des outils d'ingénierie fondés sur des logiciels échangeant des informations et des artefacts d'ingénierie le long de la chaîne d'outils liée au processus d'ingénierie.

Les systèmes de communication constituent une partie importante des systèmes techniques modernes et, en particulier, des systèmes d'automatisation qui y sont intégrés. Du fait de la décentralisation croissante des systèmes d'automatisation et de l'application des technologies de bus de terrain et d'Ethernet, il est nécessaire que la connexion des dispositifs d'automatisation et des autres entités en interaction réponde à des exigences particulières en matière de qualité, de sûreté et de sécurité des communications. Ainsi, dans le cadre du processus d'ingénierie des systèmes techniques modernes, il est nécessaire d'échanger également les informations et artefacts d'ingénierie relatifs aux systèmes de communication le long de la chaîne d'outils du processus d'ingénierie.

Chaque phase du processus d'ingénierie des systèmes techniques permet de créer des informations relatives aux systèmes de communication qui peuvent être utilisées dans les phases d'ingénierie ultérieures. Un cas d'application type est la création d'informations de configuration pour les composants de communication des dispositifs d'automatisation, y compris les adresses de communication et la structuration des paquetages de communication dans les dispositifs de programmation du contrôleur pendant la phase de programmation de commande et leur utilisation dans un outil de configuration du dispositif. Un autre cas d'application type est la transmission des configurations des dispositifs de communication à des outils virtuels de mise en service, à des outils de documentation ou à des outils de diagnostic.

À l'heure actuelle, le transfert cohérent et sans perte d'informations sur l'ingénierie des systèmes de communication tout au long de la chaîne d'ingénierie des systèmes techniques n'est pas résolu. Alors que les organisations et les entreprises utilisatrices ont fourni des formats d'échange de données pour certaines parties des informations pertinentes, comme FDCML, EDDL et GSD, les cas d'application mentionnés ci-dessus ne peuvent être couverts par un format d'échange de données. En particulier, les informations relatives aux réseaux décrivant les relations de communication et leurs propriétés et qualités ne peuvent être modélisées par un format d'échange de données.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 61131-3, *Automates programmables – Partie 3: Langages de programmation*

IEC 61131-10:2019, *Automates programmables – Partie 10: Format d'échange XML ouvert PLC*

IEC 62424:2016, *Représentation de l'ingénierie de commande de processus – Demandes sous forme de diagrammes P&I et échange de données entre outils P&ID et outils PCE-CAE*

IEC 62714-1, *Format d'échange de données techniques pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 1: Architecture et exigences générales*

IEC 62714-4, *Format d'échange de données pour une utilisation dans l'ingénierie des systèmes d'automatisation industrielle – Automation markup language – Partie 4: Logique*

IEC 81346 (toutes les parties), *Systèmes industriels, installations et appareils, et produits industriels – Principes de structuration et désignations de référence*

3 Termes, définitions, termes abrégés et acronymes

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions de l'IEC 62714-1 ainsi que les suivants, s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1.1
AutomationML/AML
format d'échange de données XML pour les données d'ingénierie d'usine

3.1.2
objet d'automatisation
entité dans un système automatisé

Note 1 à l'article: Un exemple d'objet d'automatisation est un composant d'automatisation, une vanne ou un signal.

3.2 Termes abrégés et acronymes

Pour les besoins du présent document, les abréviations suivantes s'appliquent.

AML	Automation Markup Language
CAEX	Computer Aided Engineering Exchange (échange de données techniques assistées par ordinateur), comme cela est défini dans l'IEC 62424:2016
ECAD	Computer aided engineering for electrical engineering (conception assistée par ordinateur appliquée à l'électronique)
EDDL	Electronic Device Description Language (langage de description électronique de produit)
FDCML	Field Device Configuration Markup Language (langage de balisage pour la configuration des dispositifs de terrain)
GUID	Global Unique Identifier (identifiant unique global)
GSD	General Station Description (description générale de la station)
IHM	Interface Homme Machine
ID	Identifiant

MCAD	Computer aided engineering for mechanical engineering (conception mécanique assistée par ordinateur)
OPC	Open Platform Communications
PDU	Protocol Data Unit (unité de données du protocole)
SCADA	Supervisory Control And Data Acquisition
UML	Unified Modelling Language (langage de modélisation unifié)
UUID	Universal Unique Identifier (identifiant unique universel)
XML	eXtensible Markup Language (langage de balisage extensible)

4 Cas d'utilisation et structures de réseau

4.1 Généralités

La modélisation des systèmes de communication fondée sur AutomationML vise la modélisation d'une grande quantité d'informations créées, échangées et appliquées dans le cadre du processus d'ingénierie des systèmes de fabrication. Néanmoins, toutes les informations possibles liées au système de communication ne sont pas modélisées. Dans le paragraphe suivant, les cas d'utilisation pour l'application de la modélisation du système de communication ainsi que les ensembles d'informations pertinents sont nommés.

4.2 Cas d'utilisation

4.2.1 Activités d'ingénierie

Les informations relatives à la topologie des réseaux et aux systèmes de communication sont pertinentes pour différentes activités d'ingénierie tout au long de la chaîne des systèmes de production. Dans le cadre du processus d'ingénierie des systèmes de production, le système de communication peut être conçu dans la phase d'ingénierie détaillée en utilisant différents outils. Ainsi, les informations relatives au système de communication sont créées et peuvent ensuite être appliquées dans le cadre de la conception détaillée des dispositifs et de leur mise en service. La Figure 1 représente un exemple d'ensemble d'activités d'ingénierie pertinentes dans le cadre du processus général d'ingénierie des systèmes de production et de l'ingénierie des systèmes de communication qui y est intégrée.

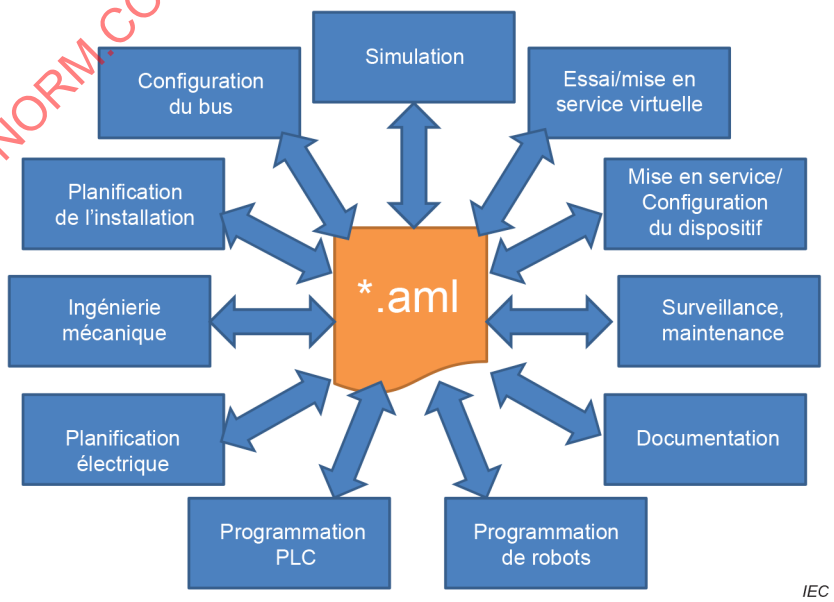


Figure 1 – Activités générales d'ingénierie dans lesquelles le système de communication est intégré

Dans le cadre des activités d'ingénierie mentionnées, des outils d'ingénierie comme les suivants (entre autres) ont un impact pertinent:

- outils de planification des installations;
- outils d'ingénierie mécanique (MCAD);
- outils d'ingénierie électrique (ECAD);
- outils de programmation du contrôleur;
- outils de programmation de robots;
- outils de programmation IHM;
- outils de configuration du système OPC;
- outils de configuration du dispositif;
- outils de configuration du bus;
- outils de simulation;
- systèmes SCADA;
- outils de mise en service virtuelle;
- outils de documentation;
- outils de sécurité des systèmes de communication;
- outils de gestion des systèmes de communication;
- outils de diagnostic des systèmes de communication.

Ces outils créent et/ou consomment des informations d'ingénierie liées au système de communication selon le cas d'utilisation de la chaîne d'ingénierie.

Néanmoins, il existe (entre autres) deux cas d'application principaux dans cette chaîne d'ingénierie, dans lesquels les informations relatives au système de communication peuvent être échangées entre les outils d'ingénierie. Ces deux cas d'utilisation sont la cible principale de la modélisation des systèmes de communication fondés sur AutomationML.

4.2.2 Transfert sans perte d'informations sur l'instance du dispositif de communication

Dans le cadre du processus général d'ingénierie, ce cas d'utilisation couvre la transition des informations de communication pertinentes pour la configuration des composants de communication des capteurs et des actionneurs depuis l'outil de programmation du contrôleur et des outils similaires vers les outils de configuration/programmation des capteurs et des actionneurs. Il contient la transition des informations pertinentes pour une communication correcte (comme les adresses et les canaux) ainsi que pour une structuration correcte des paquetages de données transmis dans le cadre de la communication (comme les points de données transmis par les capteurs). Une vue d'ensemble est présentée à la Figure 2.

Dans le cadre des activités d'ingénierie connexes, des ingénieurs ayant les rôles de programmeur de contrôleur, de programmeur d'IHM, d'ingénieur en conception électrique, de commissaire à la communication, de commissaire de commande et de programmeur de robot peuvent être impliqués. Ils exécutent la séquence suivante d'activités d'ingénierie et d'échange de données, qu'il convient de considérer comme une séquence d'exemple.

Étape 1. Conception de l'instrumentation du système, définition des dispositifs utilisés/interconnectés

Étape 2. Exportation d'informations sur le dispositif à partir de l'outil d'instrumentation du système

Étape 3. Importation d'informations sur le dispositif dans les outils de programmation et de configuration du contrôleur

Étape 4. Intégration des descriptions de dispositifs (comme la GSD) dans l'outil de programmation du contrôleur

Étape 5. Conception de programmes et de configurations de contrôleurs à l'aide d'un outil de programmation de contrôleurs

Étape 6. Exportation des informations relatives au dispositif de communication à partir de l'outil de programmation du contrôleur

Étape 7. Importation des informations relatives au dispositif de communication dans l'outil de configuration du dispositif

Étape 8. Utilisation des informations pour le paramétrage des composants de communication du dispositif (adresses, etc.) et pour la structuration des paquetages de communication (envoi de points de données)

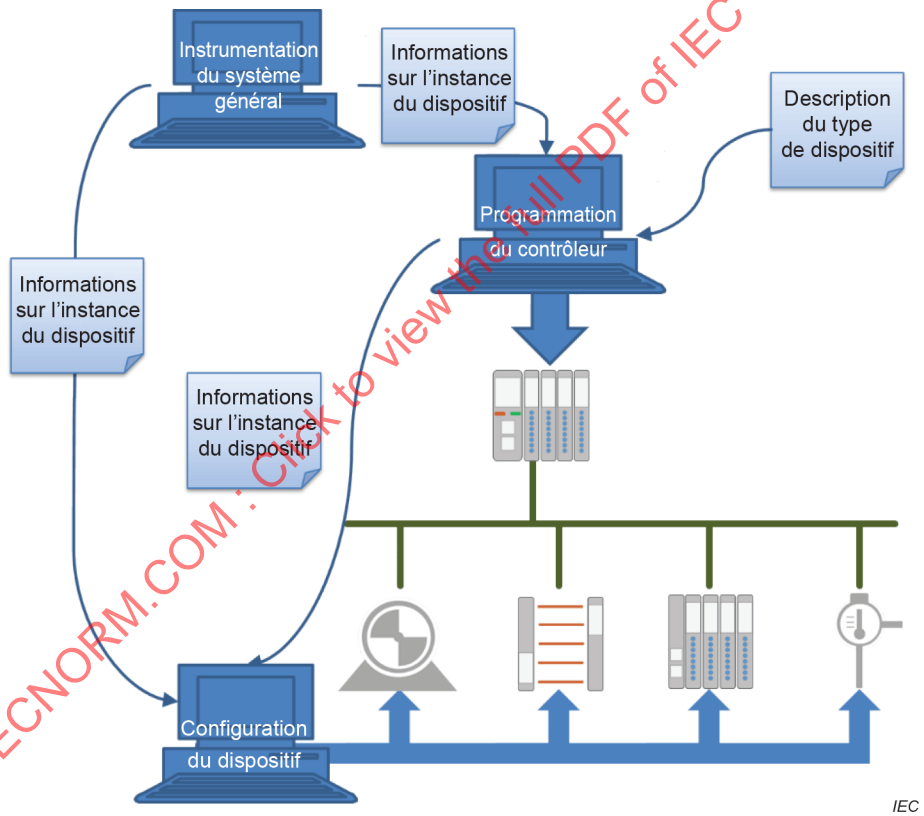


Figure 2 – Flux d'informations du cas d'utilisation

Il existe différentes alternatives possibles mais qui ne sont généralement pas appliquées. Il est possible d'imaginer les deux séquences suivantes qui sont au centre du cas d'utilisation.

Séquence 1:

Étape 1. Conception de configurations de systèmes de communication dans un outil tiers

Étape 2. Intégration des descriptions de dispositifs

Étape 3. Exportation des informations relatives au dispositif de communication

Étape 4. Importation des informations relatives au dispositif de communication dans l'outil de configuration/de programmation du dispositif

Étape 5. Utilisation des informations pour le paramétrage des composants de communication du dispositif (adresses, etc.) et pour la structuration des paquetages de communication (envoi de points de données)

Séquence 2:

Étape 1. Un outil tiers, par exemple l'outil de configuration du dispositif, fournit des informations telles que les signaux, le volume de données, décrivant les informations sur l'instance

Étape 2. Le fournisseur du dispositif fournit des descriptions du dispositif, décrivant l'information de type

Étape 3. L'outil de configuration du bus consomme ces extraits et les descriptions des dispositifs

Étape 4. Le configurateur de bus génère la configuration du bus

Étape 5. Importation de la configuration du bus dans l'outil de programmation du contrôleur

Étape 6. Utilisation d'informations provenant de périphériques à l'intérieur du programme du contrôleur

Les deux séquences sont représentées de façon schématisée à la Figure 3.

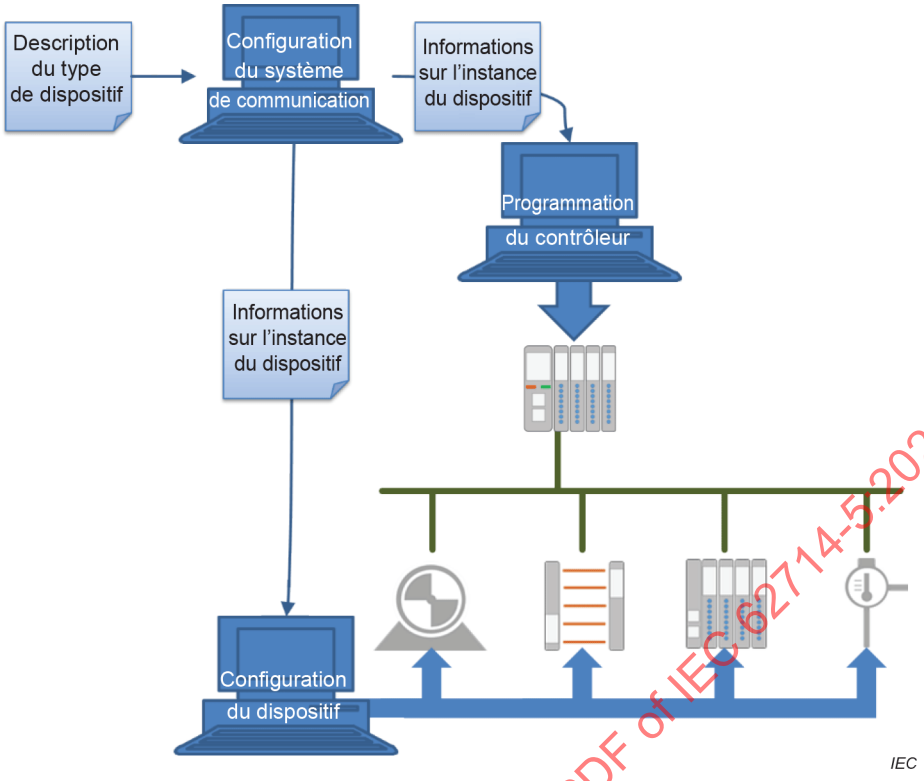


Figure 3 – Flux d'informations alternatif du cas d'utilisation

Les outils susceptibles d'exporter des informations sur le système de communication peuvent être, par exemple, des outils de programmation du contrôleur. Ils couvrent des projets de programmation de contrôleurs avec des points de données (variables), des configurations de dispositifs et des descriptions de réseaux de communication indirecte. En outre, les outils d'ingénierie des systèmes de communication peuvent être des sources de données, de même que les outils d'instrumentation comme les outils ECAD.

Les collecteurs de données de l'échange de données peuvent être des outils de configuration de la communication avec les capteurs, de programmation de l'IHM, de programmation du robot, de configuration du système OPC ou de configuration de la communication avec les acteurs. Ils couvrent principalement des projets de programmation avec des points de données (variables), des configurations de dispositifs et des descriptions de réseaux de communication indirecte. Les outils pertinents peuvent être des outils FDT, des outils de programmation IHM et des outils de programmation de robots.

À partir de ce cas d'utilisation, il convient que la modélisation des systèmes de communication fondée sur AutomationML couvre les informations relatives aux listes d'E/S, à l'association des variables (E/S) aux paquetages des données de communication et aux paramètres des dispositifs pour le paramétrage des dispositifs de communication tels que les adresses (par exemple, l'adresse IP), l'accès au support (par exemple, l'adresse MAC), les masques de sous-réseau, les adresses de passerelle et les objets de communication utilisés, tels que les informations de profil.

En outre, il convient que la modélisation réponde aux exigences non fonctionnelles suivantes. La liste des paramètres du dispositif doit pouvoir être étendue par les utilisateurs pour couvrir les technologies à venir. Des RoleClassLibs et/ou SystemUnitClassLibs appropriées permettant l'identification de la sémantique des objets doivent être définies.

4.2.3 Transfert sans perte d'informations sur les systèmes de communication

Au-delà de la configuration des dispositifs de communication, les informations du système de communication sont exploitées dans différents outils d'ingénierie, de surveillance, de maintenance, etc. Par conséquent, ce cas d'utilisation couvre la transmission d'informations sur la configuration et la structure du réseau de communication, y compris la configuration des dispositifs d'infrastructure, la configuration des dispositifs d'extrémité en ce qui concerne les paramètres du système de communication, la structure et le câblage du réseau, la qualité du service, etc. Ces informations peuvent être fournies aux outils de configuration des dispositifs, aux outils de documentation et de maintenance et aux outils de gestion du réseau. Ils peuvent permettre la combinaison du câblage physique avec des connexions de communication logiques pour la détection des erreurs.

Ce cas d'utilisation fait intervenir des ingénieurs ayant les rôles suivants: programmeur de contrôleur, programmeur d'IHM, ingénieur en conception électrique, commissaire à la communication, commissaire au contrôle, programmeur de robot et opérateur. Ils exécutent le processus d'ingénierie suivant, qu'il convient de considérer comme une séquence d'exemple.

Étape 1. Ingénierie des ensembles de variables à transmettre au sein du système de communication dans les outils de programmation du contrôleur et les outils de programmation/configuration des capteurs/actionneurs

Étape 2. Exportation de ces ensembles de variables à partir des outils de programmation des contrôleurs et des outils de programmation/configuration des capteurs/actionneurs

Étape 3. Importation d'ensembles de variables dans les outils d'ingénierie des réseaux de communication

Étape 4. Ingénierie de la mise en correspondance des listes de variables et ingénierie du câblage physique

Étape 5. Exportation de la conception des systèmes de communication à partir d'outils d'ingénierie des réseaux de communication

Étape 6. Importation de la conception des systèmes de communication vers des outils d'ingénierie ou d'exécution continue

La séquence est représentée de façon schématisée à la Figure 4.

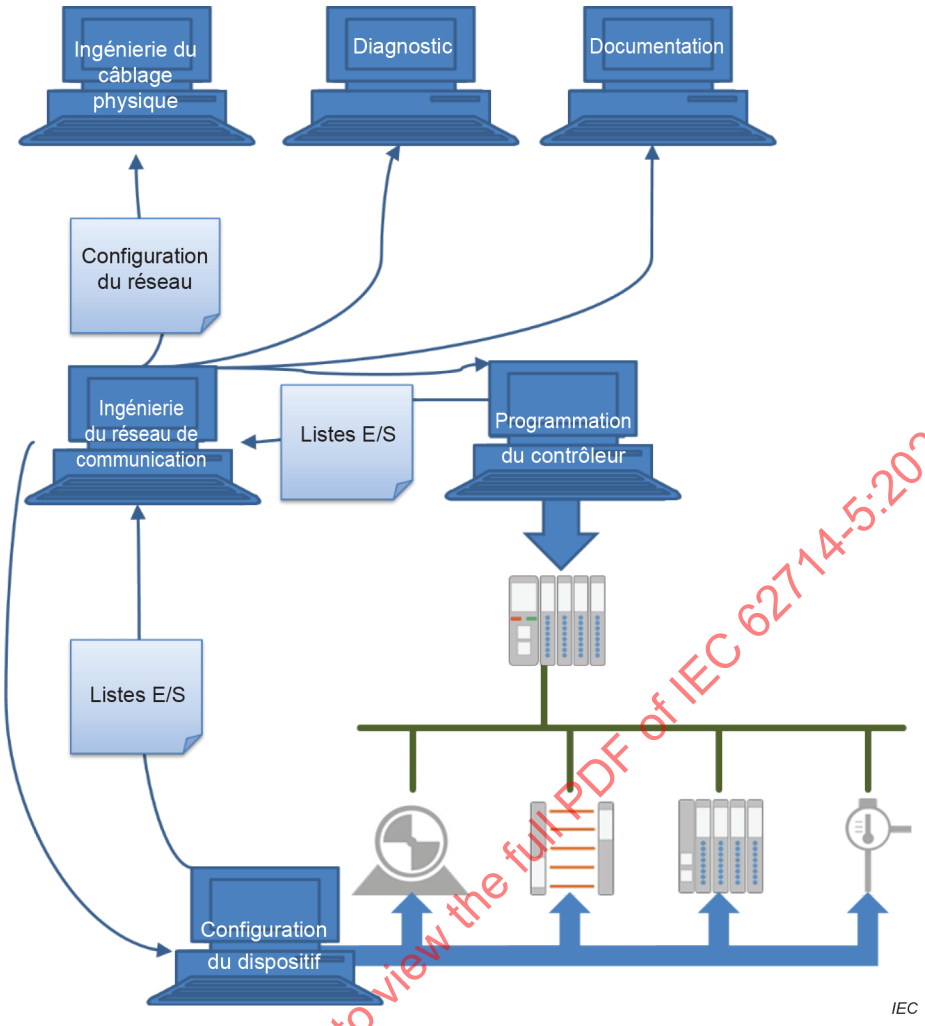


Figure 4 – Flux d'informations du cas d'utilisation

Les sources des données d'ingénierie échangées peuvent être des outils de programmation de contrôleur couvrant des projets de programmation de contrôleur avec des points de données (variables), des configurations de dispositif et des descriptions de réseau de communication indirect, des outils d'ingénierie de système de communication qui couvrent des projets de système de communication comprenant des configurations de dispositif et des propriétés de connexion de communication, des outils de configuration de communication de capteur, des outils de programmation d'IHM, des outils de programmation de robot, des outils de configuration de système OPC ou des outils de configuration de communication d'acteur qui couvrent tous ensemble des projets de programmation avec des points de données (variables), des configurations de dispositif et des descriptions de réseau de communication indirect.

Les collecteurs de l'échange de données peuvent être des outils de surveillance de la sécurité des systèmes de communication, de configuration des systèmes de communication, de gestion des systèmes de communication et de diagnostic des systèmes de communication.

À partir de ce cas d'utilisation, la modélisation des systèmes de communication fondée sur AutomationML doit couvrir les informations relatives à la topologie du réseau, aux listes de connecteurs des dispositifs et à leurs propriétés, aux connexions physiques entre les connecteurs des dispositifs et à leurs propriétés, aux liens logiques entre les applications fondées sur les dispositifs et à leurs propriétés, à la mise en correspondance des liens logiques aux connexions physiques, à l'association des attributs des liens et/ou des connexions aux liens ou aux connexions comme les adresses des émetteurs et des récepteurs, la qualité du service, les protocoles utilisés, etc.

En outre, certaines exigences non fonctionnelles sont dérivées. La liste des attributs de lien et de connexion doit pouvoir être étendue par les utilisateurs pour couvrir les technologies à venir. Des RoleClassLibs et/ou SystemUnitClassLibs appropriées permettant l'identification de la sémantique des objets doivent être définies.

4.3 Délimitation de la plage de modélisation

4.3.1 Domaine d'application de la plage de modélisation

Dans le cadre des cas d'utilisation cités ci-dessus, différentes technologies de communication, stratégies d'utilisation et cycles de vie des systèmes de communication ainsi que des architectures de réseaux de communication peuvent être pris en considération. Dans ce qui suit, la large plage théorique est limitée à la plupart des cas appliqués.

4.3.2 Structures d'interaction et cycles de vie

Conformément aux cas d'utilisation décrits ci-dessus, il convient que la modélisation du système de communication fondée sur AutomationML se concentre sur la description de la communication cyclique fondée sur une attribution fixe entre les partenaires de communication. Ces partenaires de communication exécutent des parties (plus ou moins complexes) de la logique de commande des applications de contrôle sur différents dispositifs de contrôle communicants.

Par conséquent, il existe au moins deux vues différentes concernant le système de communication: une vue logique et une vue physique. D'un point de vue logique de haut niveau, les applications de traitement des données échangent des données d'application (variables) de manière logique. Pour la réalisation physique des applications de traitement des données, des dispositifs de contrôle sont utilisés. Ces dispositifs communiquent physiquement à l'aide de systèmes de communication. Dans les deux cas, les objets modélisés (parties de l'application, dispositifs, connexions, etc.) ont des propriétés spéciales qui sont importantes pour l'utilisation/le comportement corrects du système de communication. Ces systèmes et leurs informations descriptives font l'objet des présentations suivantes.

La modélisation des systèmes de communication fondée sur AutomationML ne concerne pas, par exemple, les éléments suivants:

- des structures de communication acycliques fondées sur les services et utilisées uniquement de manière temporaire, telles qu'elles sont généralement présentes dans l'utilisation du World Wide Web;
- les questions liées à la sûreté ou à la sécurité; et
- les informations complexes sur le comportement et la structure des dispositifs, car elles sont généralement couvertes par les langages de description des dispositifs.

Néanmoins, l'utilisation d'AutomationML pour modéliser des systèmes de communication ayant ces propriétés n'est pas exclue, bien que non prévue.

Les descriptions des systèmes de communication peuvent couvrir différents niveaux de détail. Il peut s'agir de descriptions très abstraites ne désignant que les partenaires de communication et leurs dépendances, ou d'une description détaillée des connexions de communication et de leurs propriétés, ainsi que des parties de dispositifs et d'applications avec leurs propriétés. Ces différents niveaux de détail entrent dans le domaine de la modélisation du système de communication envisagé.

Les systèmes de communication sont des systèmes complexes qui couvrent différentes technologies au sein d'un même système de communication. À titre d'exemple, les systèmes de contrôle fondés sur des contrôleurs peuvent être pris en considération. Ils intègrent généralement différents systèmes de bus de terrain, une communication fondée sur Ethernet, un câblage en cuivre et en fibre optique, une communication sans fil, une infrastructure active différente, etc. Différents protocoles de communication peuvent fonctionner sur la même infrastructure physique (fil, canal sans fil, etc.). Un seul dispositif peut exécuter plusieurs parties d'application de contrôle. Ainsi, les différentes technologies de communication sont liées entre elles et dépendent les unes des autres. Il convient que cette complexité et cette interconnexion puissent être exprimées par AutomationML et soient dans le domaine d'application des descriptions suivantes.

Les systèmes de communication sont pris en considération dans différentes phases de l'ingénierie d'un système de production, telles que l'ingénierie détaillée et la mise en service, et sont utilisés dans différentes phases du cycle de vie d'un système de production, telles que la configuration du dispositif, la rampe croissante de la production, le déroulement normal de la production, le diagnostic et la maintenance. Il convient de couvrir ces différentes phases d'ingénierie et d'utilisation des systèmes de production et leur impact sur le contenu et l'utilisation du modèle par la modélisation des systèmes de communication fondés sur AutomationML.

4.3.3 Objets du réseau

Chaque réseau de communication est considéré comme un réseau à deux niveaux: un niveau logique et un niveau physique. La mise en correspondance de ce point de vue avec la structure de couche ISO OSI 7 des systèmes de communication [2][1], souvent citée en référence, est la suivante: le niveau physique couvre les couches 1 et 2 de la structure de couche ISO OSI 7, tandis que le niveau logique englobe les couches 3 à 7 de la structure de couche ISO OSI 7 et les applications de contrôle situées au-dessus de la couche 7.

En regardant la couche logique, les entités suivantes à modéliser peuvent être trouvées. Les applications de contrôle se composent généralement de plusieurs parties d'applications distribuées qui fournissent différentes fonctionnalités du processus de contrôle et qui forment des dispositifs logiques. Un exemple (donné à la Figure 5) peut être une application de contrôle principale échangeant des informations sur les capteurs et les actionneurs avec deux fonctions d'entrée/sortie. En général, ces dispositifs logiques (parties de l'application de contrôle) échangent des informations de différents types, dans l'exemple différents signaux de capteurs et états d'actionneurs, qui peuvent être considérés comme des points de connexion aux dispositifs logiques et des points d'extrémité de l'échange d'informations entre les dispositifs logiques. L'échange d'informations lui-même est exécuté par différentes connexions logiques.

[1] Les chiffres entre crochets renvoient à la Bibliographie.

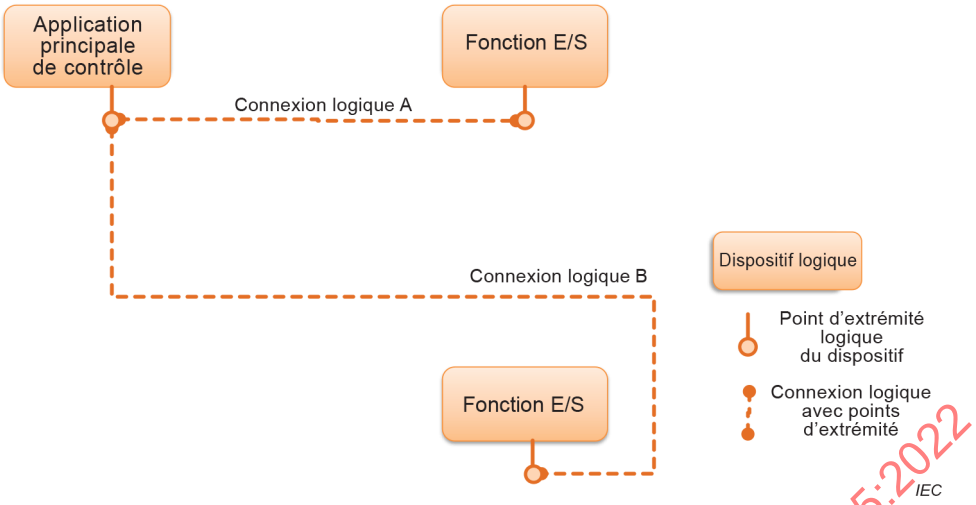


Figure 5 – Exemple d'une vue de niveau logique sur les systèmes de communication

Le réseau logique peut contenir différents objets ayant des propriétés descriptives différentes. Alors que les dispositifs de contrôle logiques peuvent avoir des identifiants, des temps de cycle et des empreintes uniques (pour ne citer que quelques exemples), les points d'extrémité logiques peuvent avoir un type de données et les connexions logiques peuvent avoir un taux de transmission exigé. Dans tous les cas, ces propriétés descriptives peuvent être considérées comme des attributs des objets d'intérêt.

Au niveau physique, se trouvent des dispositifs physiques avec des interfaces physiques. Dans l'exemple donné à la Figure 6, un contrôleur est connecté à deux dispositifs d'E/S par l'intermédiaire d'une infrastructure active. Les dispositifs physiques ont des extrémités physiques qui représentent des interfaces de réseau comme des fiches et des prises et sont reliés par des connexions physiques à un système de communication. Par rapport à la vue du niveau logique, il existe des entités physiques supplémentaires qui représentent les composants de l'infrastructure du réseau (comme les commutateurs, etc.).

Comme dans le cas du réseau logique, les objets du réseau physique peuvent avoir des propriétés descriptives différentes. Alors que les dispositifs physiques peuvent avoir des identifiants et des capacités de traitement, les points d'extrémité physiques peuvent avoir une adresse, un débit de données maximal et un type de connecteur, tandis que les connexions physiques peuvent avoir un débit de transmission possible, un type de fil, un type d'antenne, un type de fiche ou un numéro de documentation. Ces propriétés descriptives peuvent être considérées comme des attributs des objets d'intérêt.

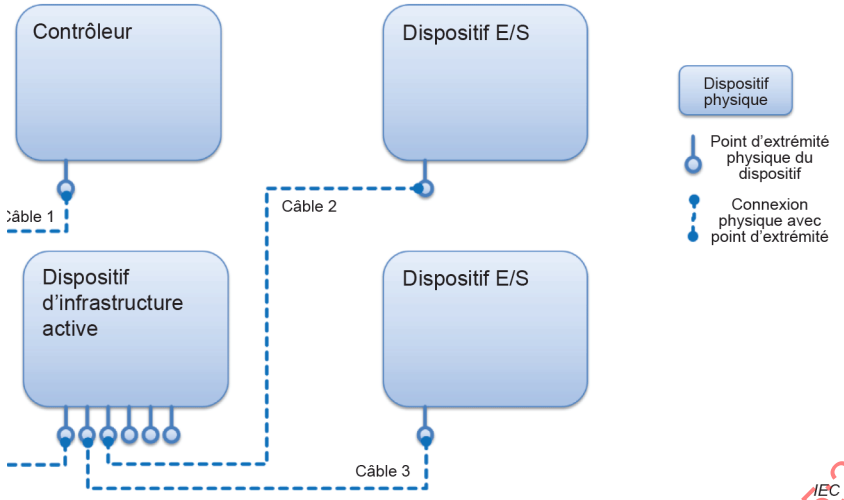


Figure 6 – Exemple d'une vue de niveau physique sur les systèmes de communication

Il est nécessaire de combiner les deux vues pour obtenir la description complète du système de communication. Par conséquent, les dispositifs logiques sont hébergés par des dispositifs physiques interconnectés dans une topologie de réseau physique. Dans l'exemple donné à la Figure 7, un contrôleur héberge une application de contrôle principale tandis que les dispositifs E/S hébergent les fonctions d'E/S. En outre, les interfaces physiques et les interfaces logiques sont mises en correspondance. Ainsi, chaque connexion logique est mise en correspondance virtuelle avec un ensemble de connexions physiques qui la mettent en œuvre. Il n'est pas strictement nécessaire qu'il existe une chaîne unique de connexions physiques représentant cette mise en œuvre, car tous les systèmes de communication ne le permettent pas.

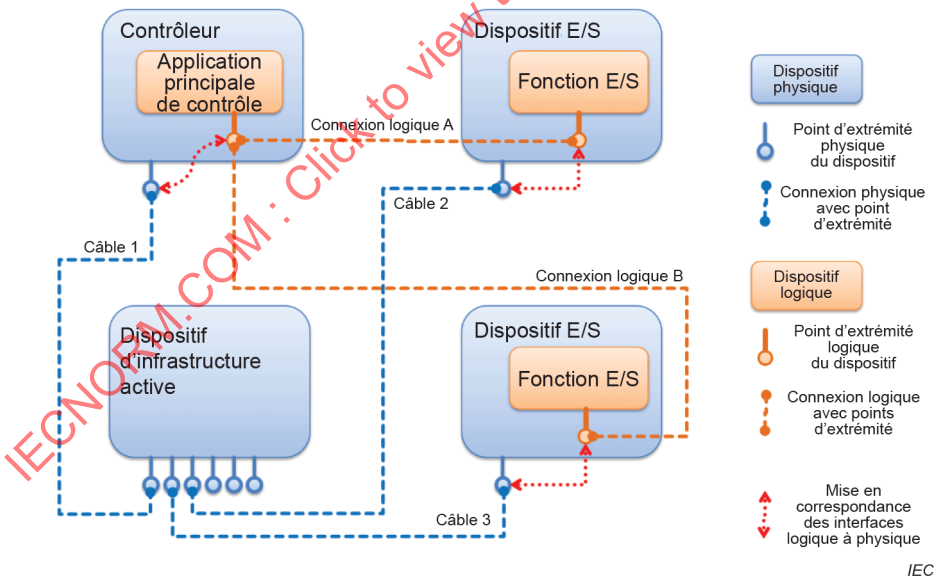


Figure 7 – Vues combinées sur les systèmes de communication

4.3.4 Topologies de réseau

Dans les systèmes de communication industriels, différentes architectures de réseau doivent être prises en considération. D'une part, il est possible de distinguer différentes topologies de câblage. D'autre part, différentes topologies de réseau physique sont possibles, elles dépendent souvent du routage des câbles ou des paramètres des canaux sans fil. Il convient que la modélisation des systèmes de communication présentée ici soit applicable à tous ces systèmes.

En ce qui concerne les différentes topologies de câblage, il convient que la modélisation puisse couvrir les topologies en étoile, linéaires et en anneau, y compris leurs combinaisons. Des exemples sont donnés à la Figure 8, à la Figure 9 et à la Figure 10.

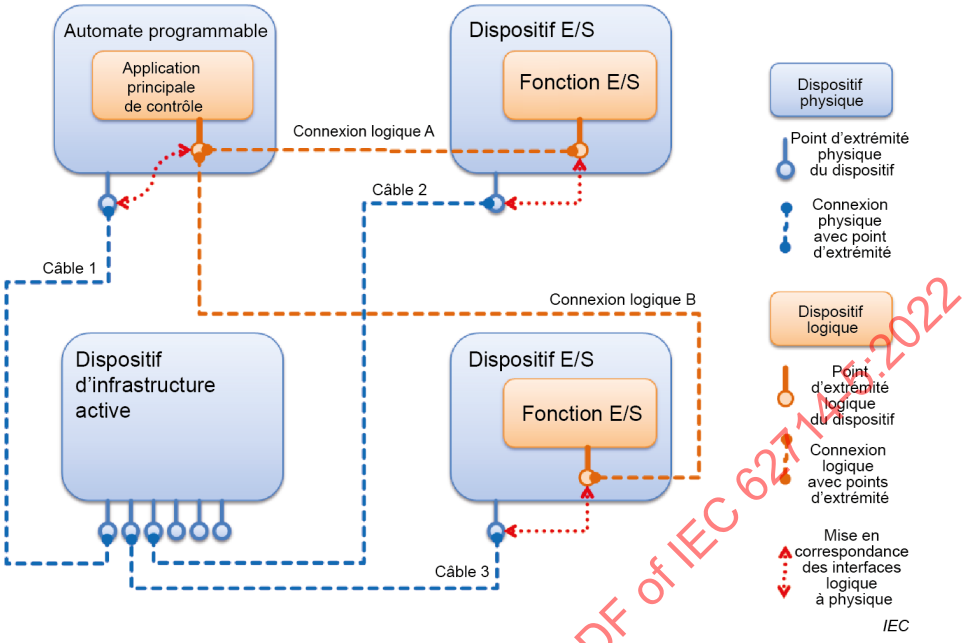


Figure 8 – Exemple de topologie en étoile

La Figure 8 représente une topologie en étoile type dans laquelle le dispositif d'infrastructure active établit des connexions entre les différents dispositifs physiques tandis que les connexions logiques restent des connexions bilatérales entre dispositifs logiques.

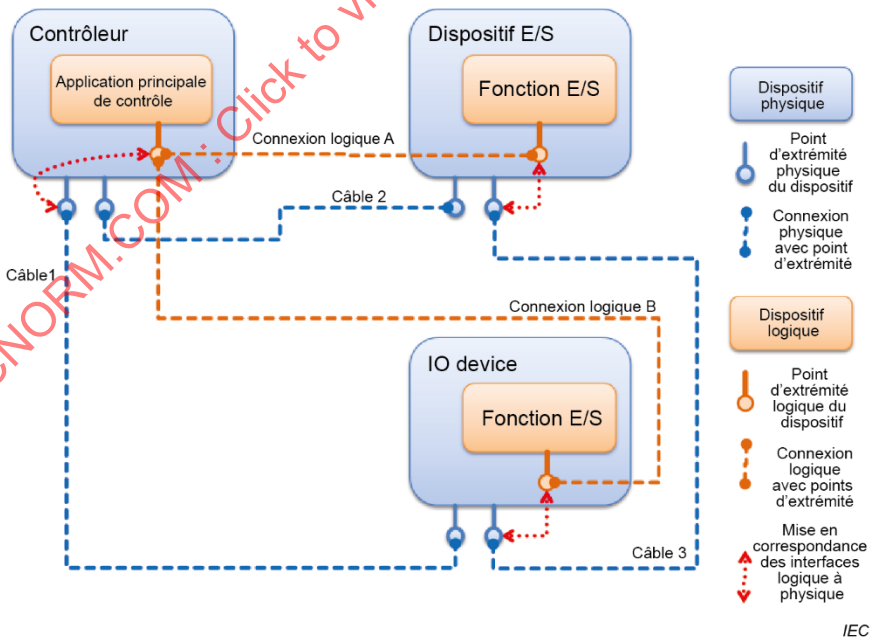


Figure 9 – Exemple de topologie en anneau

La Figure 9 représente une topologie en anneau type dans laquelle chaque dispositif physique comporte deux interfaces physiques qui relient les dispositifs dans un anneau physique. Là encore, les connexions logiques restent des connexions bilatérales entre les dispositifs logiques.

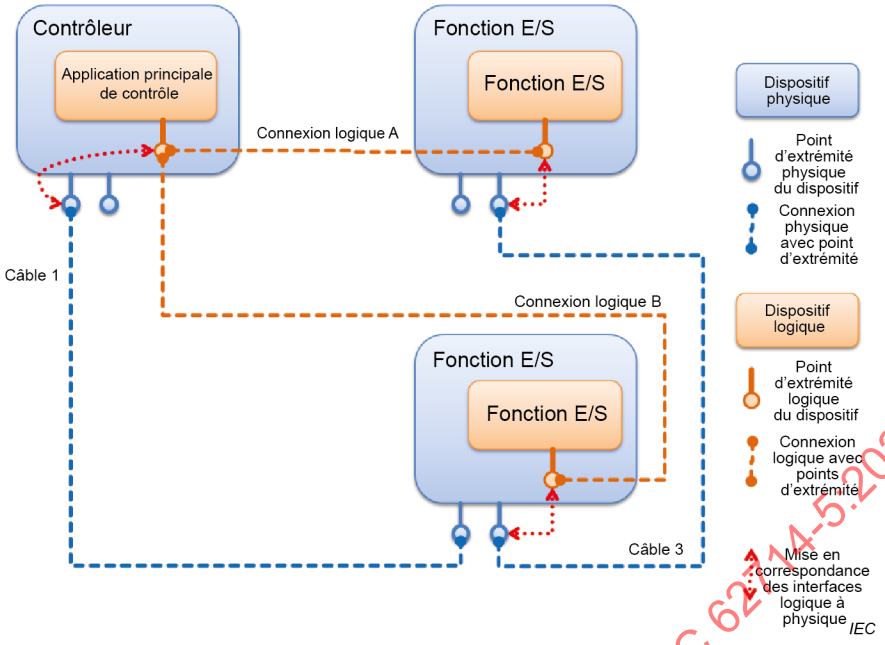


Figure 10 – Exemple de topologie linéaire

La Figure 10 représente une topologie linéaire type. Là encore, chaque dispositif physique comporte deux interfaces physiques qui relient les dispositifs dans une ligne physique. Il s'agit donc d'une topologie en anneau ouvert. Les connexions logiques restent des connexions bilatérales entre les dispositifs logiques.

En ce qui concerne les topologies de réseau, il convient que la modélisation permette de représenter des réseaux simples de câblage direct de composants, des réseaux avec des infrastructures actives, des réseaux connectés par des passerelles et des réseaux hiérarchiques (c'est-à-dire des réseaux dans lesquels un composant net contient un autre réseau complet, comme un bus de fond de panier dans un contrôleur modulaire). Des exemples de telles structures de réseau sont donnés à la Figure 11, à la Figure 12, à la Figure 13 et à la Figure 14.

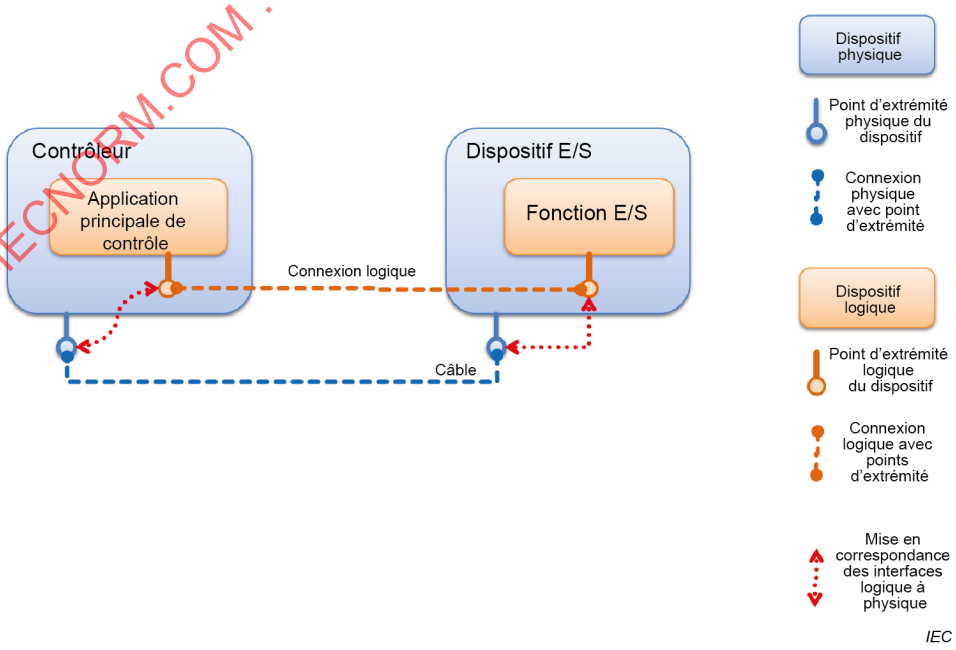


Figure 11 – Réseau simple avec câblage direct

La Figure 11 représente un réseau très basique dans lequel deux dispositifs physiques sont directement câblés en établissant une connexion logique entre les dispositifs logiques correspondants. Il peut être considéré comme le réseau minimal qui doit être représentable.

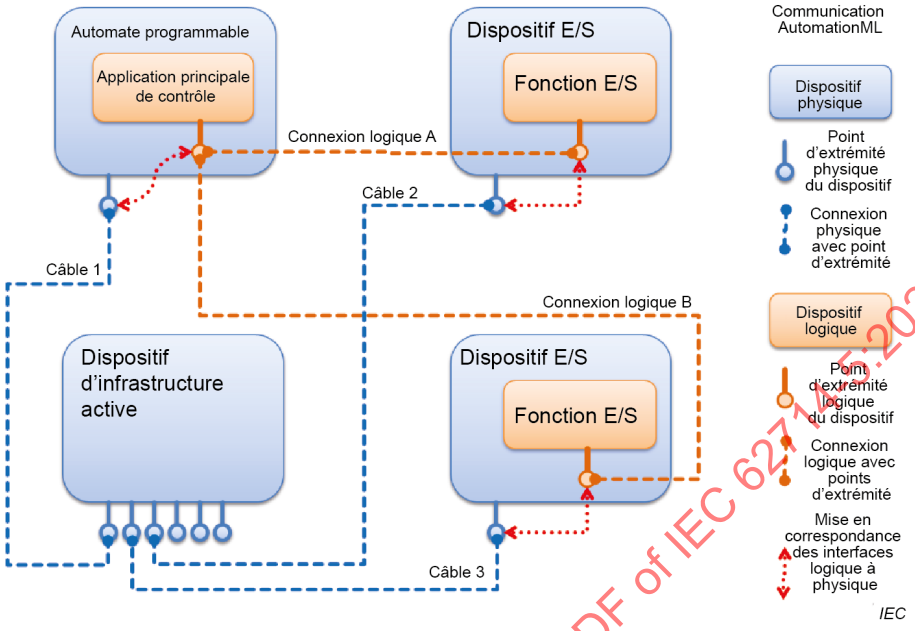


Figure 12 – Réseau avec infrastructure active

La Figure 12 représente un réseau qui contient des dispositifs d'infrastructure actifs, comme c'est généralement le cas dans les réseaux fondés sur Ethernet dans lesquels des commutateurs, des concentrateurs et des routeurs sont intégrés pour transmettre la communication.

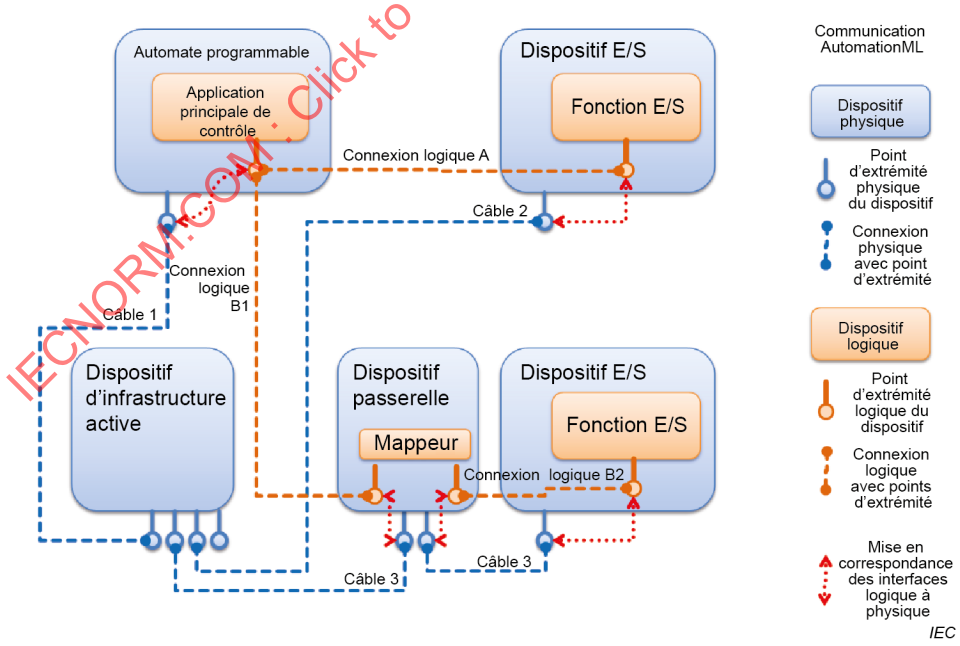


Figure 13 – Réseaux connectés par des passerelles

La Figure 13 représente un réseau dans lequel une connexion logique s'étend sur deux réseaux différents en exploitant une passerelle d'application pour la mise en correspondance des connexions logiques. Cette figure indique qu'au sein de cette passerelle, les éléments de données des deux connexions pertinentes sont mis en correspondance.

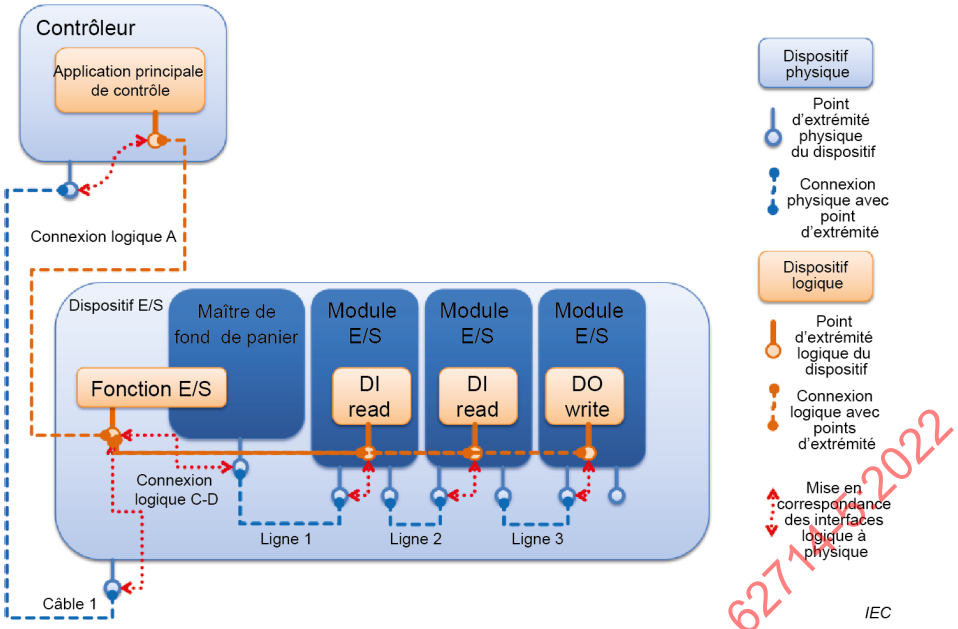


Figure 14 – Réseaux structurés hiérarchiques

La Figure 14 représente une structure de réseau hiérarchique telle qu'elle est généralement donnée dans le cas d'un système de communication qui contient un dispositif physique avec un réseau de fond de panier qui relie les différents composants internes du dispositif physique. Ce cas est donné lorsque des coupleurs de bus de terrain modulaires sont exploités dans un réseau.

En plus de l'application possible de différentes topologies et structures d'interaction de réseau dans les systèmes de communication industriels, il existe généralement différentes applications connectées par un système de communication, comme le contrôle et la configuration/programmation du système. Il convient de modéliser également les réseaux couvrant plusieurs applications comme cela est représenté à la Figure 15.

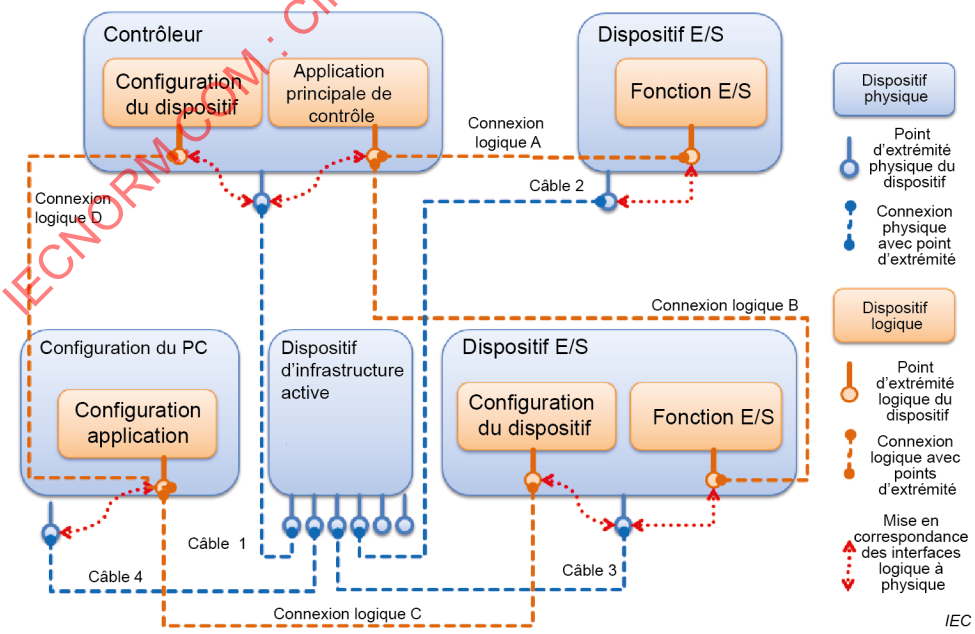


Figure 15 – Réseau couvrant plusieurs applications

4.3.5 Contenu de la communication

Dans les systèmes de communication, les datagrammes de communication (appelés unités de données de protocole / PDU - *protocol data units*) sont échangés entre les partenaires de communication, c'est-à-dire les parties d'application de contrôle. Par conséquent, chaque connexion logique contient des objets PDU échangés par l'intermédiaire de cette connexion. Dans le cas d'un objet PDU contenant des informations de commande (signaux des capteurs et des acteurs, état, alarmes, etc.), il peut être lié à une PLCopenXMLInterface (voir l'IEC 61131-10) ou à une SignalInterface modélisant les informations échangées.

Cette structure est décrite à la Figure 16.

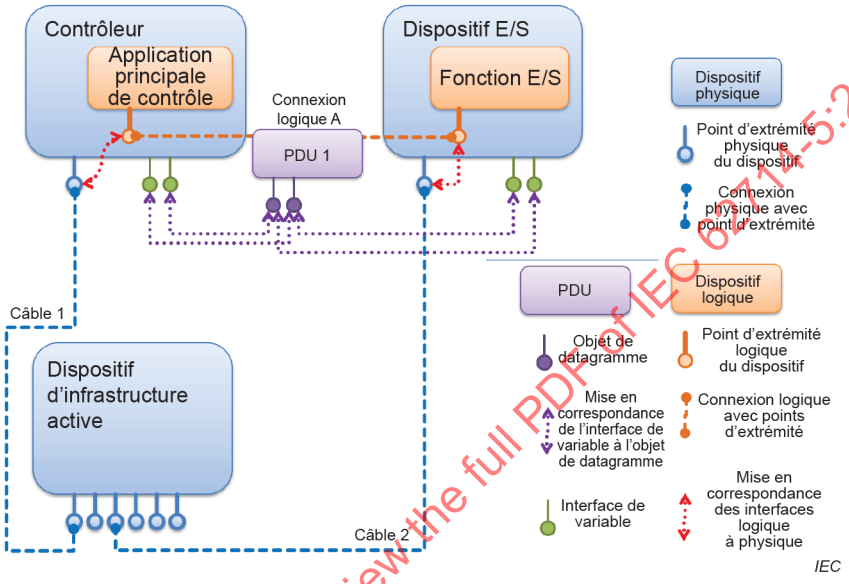


Figure 16 – Stratégie générale de modélisation des PDU

4.4 Exigences de modélisation dérivées

À partir des observations données dans le présent paragraphe, les informations générales suivantes sont modélisées:

a) informations de niveau physique
   1) dispositifs physiques qui représentent les entités physiques communicantes avec leurs propriétés;
   2) points d'extrémité des connexions physiques fixées à des dispositifs physiques qui représentent l'interface physique des dispositifs (prise) avec leurs propriétés descriptives;
   3) connexions physiques qui représentent le câblage physique ou les canaux sans fil utilisés avec leurs propriétés descriptives.

b) informations de niveau logique
   1) dispositifs logiques qui représentent les parties de l'application avec leurs propriétés descriptives;
   2) points d'extrémité des connexions logiques fixées aux dispositifs logiques qui représentent l'interface de la partie de l'application avec leurs propriétés descriptives;
   3) connexions logiques qui représentent l'échange d'informations entre les parties de l'application avec leurs propriétés descriptives;
   4) les unités de données de traitement modélisent les paquetages de données échangés.

c) relations entre les niveaux logique et physique
   1) mise en correspondance des points d'extrémité pour représenter la mise en œuvre des connexions logiques par des connexions physiques;

2) relations entre les variables échangées et les technologies/protocoles de communication appliqués, notamment par la mise en correspondance des variables avec les structures des paquetages de communication.

d) réseaux de communication

1) association unique des dispositifs et des connexions aux réseaux;
2) structuration hiérarchique unique du réseau;
3) dépendances entre les réseaux;
4) dépendances entre les dispositifs, telles que les propriétés de redondance;
5) dépendances entre les connexions, telles que les propriétés de redondance.

5 Modèle de UML

5.1 Vue d'ensemble

Le modèle UML utilise des diagrammes de classe qui représentent les relations statiques entre les entités du réseau. Les entités sont toujours décrites comme des types. La mise en œuvre réelle du modèle dépend des préférences de l'outil. Une agrégation démontre dans le cas présent une association "has a" ou une relation entre les classes et l'agrégation est plus spécifique que l'association [1].

Les cardinalités définies sont plus importantes dans les diagrammes de classe UML. Si la cardinalité n'est pas définie, elle doit être lue comme 0...n.

La Figure 17 indique les points centraux de la structure de communication modélisée d'un seul réseau de communication. Le dispositif physique est l'objet central des descriptions ultérieures. En particulier, la structure modélisée est divisée en vues topologiques logiques et physiques.

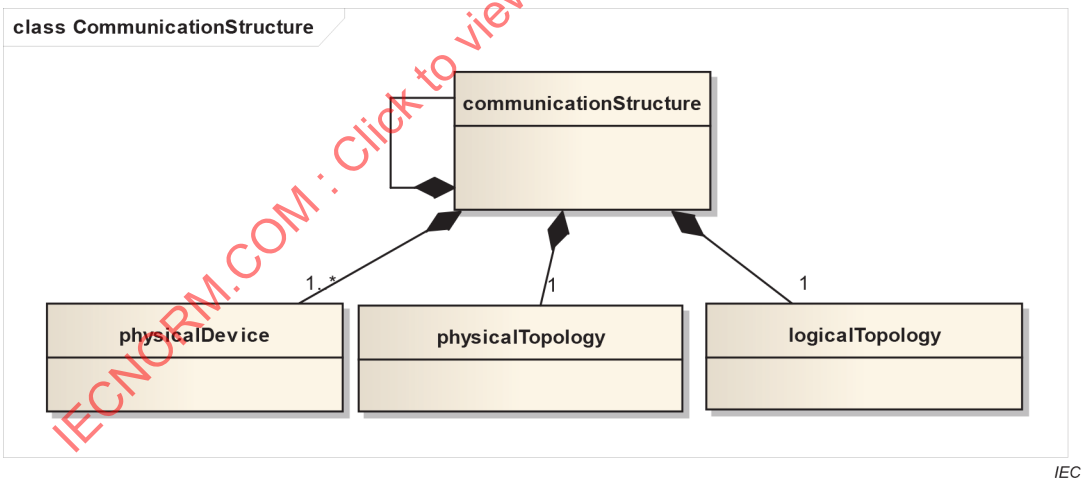


Figure 17 – Structure d'un réseau de communication

La relation de la structure de communication avec elle-même (association réflexive) permet la représentation de réseaux de communication imbriqués ou la représentation de plusieurs segments de réseau interconnectés.

5.2 Topologie logique

5.2.1 Objectif de la topologie logique

La topologie logique décrit la vue logique des partenaires de communication. Elle représente les relations de communication du point de vue des programmes d'application. La Figure 18 représente les trois éléments (abstraits) concernés et leurs relations:

- logicalTopology;
- logicalConnection;
- logicalEndPoint.

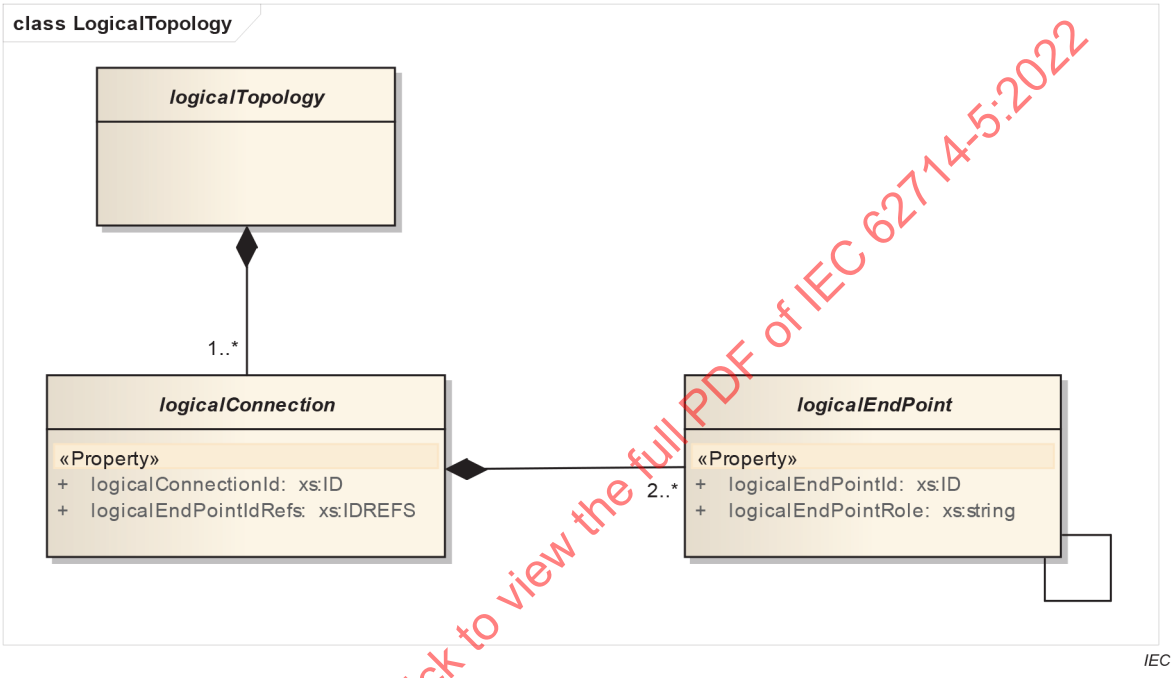


Figure 18 – Vue de la topologie logique

5.2.2 Élément logicalTopology

La classe logicalTopology permet de modéliser un réseau d'automatisation ou de communication au-dessus d'un réseau physique. Elle doit regrouper toutes les relations de communication entre les points d'extrémité logiques, qui sont à leur tour associés à des dispositifs physiques.

Une dérivation réelle de logicalTopology peut être la représentation d'un réseau de contrôle dans une installation.

5.2.3 Élément logicalConnection

L'entité logicalConnection doit décrire la relation de communication entre deux points d'extrémité logiques, qui sont des partenaires de communication. Un terme plus générique est "lien logique", qui indique la relation entre au moins deux logicalEndPoints.

La propriété logicalConnectionId est un GUID dans communicationStructure et doit identifier logicalConnection.

La propriété logicalEndPointIdRefs est un tableau et doit contenir tous les GUID des logicalEndPoints liés à logicalConnection. Ainsi, il est possible de modéliser des relations m:n dans une relation de communication.

Une dérivation réelle de logicalConnection peut être une relation de connexion dans une relation de pair à pair.

5.2.4 Élément logicalEndPoint

L'entité logicalEndPoint doit représenter un partenaire de communication dans une telle relation.

La propriété logicalEndPointId est un GUID dans communicationStructure et doit identifier logicalEndPoint.

La propriété logicalEndPointRole est de type chaîne et doit être utilisée pour spécifier le rôle logique de logicalEndPointId dans la relation de communication. Les valeurs concevables peuvent être des paires telles que "source" – "destination", "source" – "collecteur", "éditeur" – "abonné" ou "maître" – "esclave", etc.

Les dérivés d'un tel point d'extrémité de communication peuvent être un port IP ou un pair.

5.3 Topologie physique

5.3.1 Objectif de la topologie physique

La topologie physique décrit la vue physique des partenaires de communication. Elle doit représenter les relations de communication du point de vue du câblage ou des réglages des canaux. La Figure 19 représente les trois éléments (abstraits) concernés et leurs relations:

- physicalTopology;
- physicalConnection;
- physicalEndPoint.

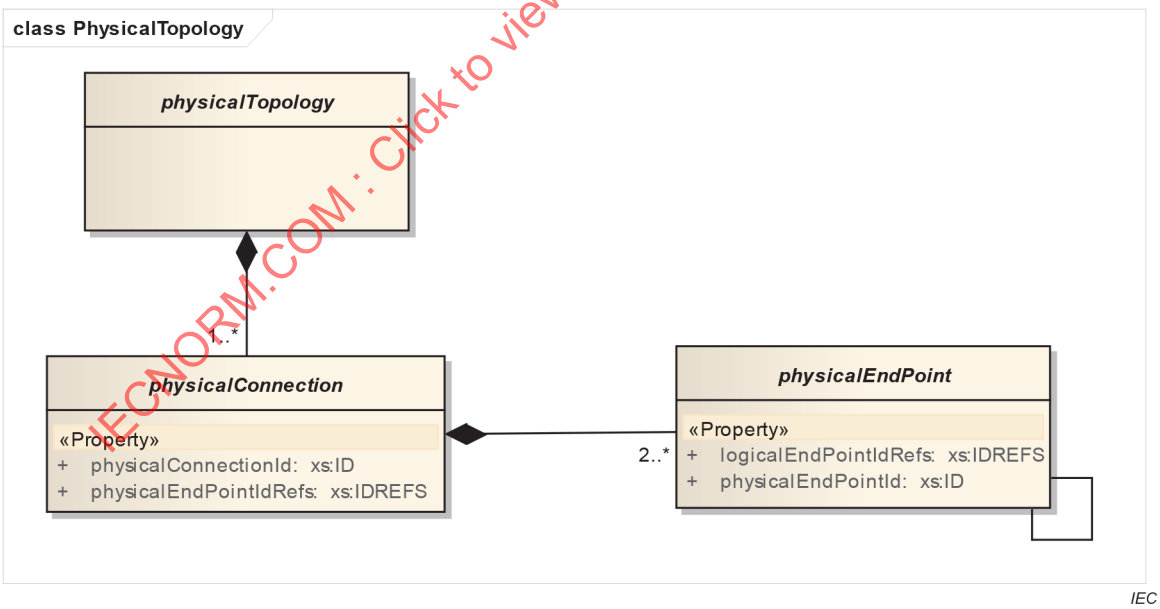


Figure 19 – Vue de la topologie physique

5.3.2 Élément physicalTopology

La classe physicalTopology est le point d'entrée pour la modélisation de la topologie physique d'un réseau d'automatisation. Elle doit regrouper toutes les relations de communication entre les points d'extrémité physiques, associés à des dispositifs d'automatisation.

Une dérivation réelle de physicalTopology peut être la représentation de réseaux physiques dans une installation.

5.3.3 Élément physicalConnection

L'entité physicalConnection doit décrire la relation de communication entre deux points d'extrémité physiques. Les physicalEndPoints peuvent également être connectés au moyen d'un réseau sans fil. Dans ce cas, un lien au moyen de l'éther est établi.

La propriété physicalConnectionId est un GUID dans communicationStructure et doit identifier physicalConnection.

La propriété physicalEndPointIdRefs est un tableau et doit contenir tous les GUID des physicalEndPoints liés à physicalConnection. Ainsi, il est possible de modéliser des connexions m:n dans un réseau de communication.

Une dérivation réelle de physicalConnection peut être un câble reliant des prises dans une relation de pair à pair.

5.3.4 Élément physicalEndPoint

L'entité physicalEndPoint doit représenter un point d'extrémité de communication réel dans une telle relation.

La propriété physicalEndPointId est un GUID dans communicationStructure et doit identifier physicalEndPoint.

La propriété logicalEndPointIdRefs est un tableau et doit contenir les GUID des logicalEndPoints liés à ce physicalEndPoint.

La dérivation d'un tel point d'extrémité peut être un port Ethernet.

5.4 Dispositif

5.4.1 Généralités

Le modèle du dispositif est le plus complexe du modèle UML. La Figure 20 en représente la première partie et la Figure 21 la deuxième partie.

5.4.2 Élément physicalDevice

Le composant central est physicalDevice. Les attributs permettent d'identifier de manière unique les instances et les sous-instances de physicalDevice. En outre, il existe une relation d'agrégation avec lui-même. Cette relation permet de modéliser des dispositifs modulaires. L'agrégation de l'élément d'information résume toutes les descriptions importantes des données du produit, les identifiants, la documentation, etc. sous la forme d'un objet central, qui peut également être utilisé par les instances d'un logicalDevice. Un physicalDevice ou ses sous-modules regroupent les logicalDevices. En outre, les points d'extrémité physiques, par exemple les ports, sont attribués au physicalDevice au moyen de physicalEndPointList. Le même schéma est utilisé pour les canaux physiques. Dans le cas où physicalDevice est un contrôleur, il peut contenir plusieurs deviceResources.

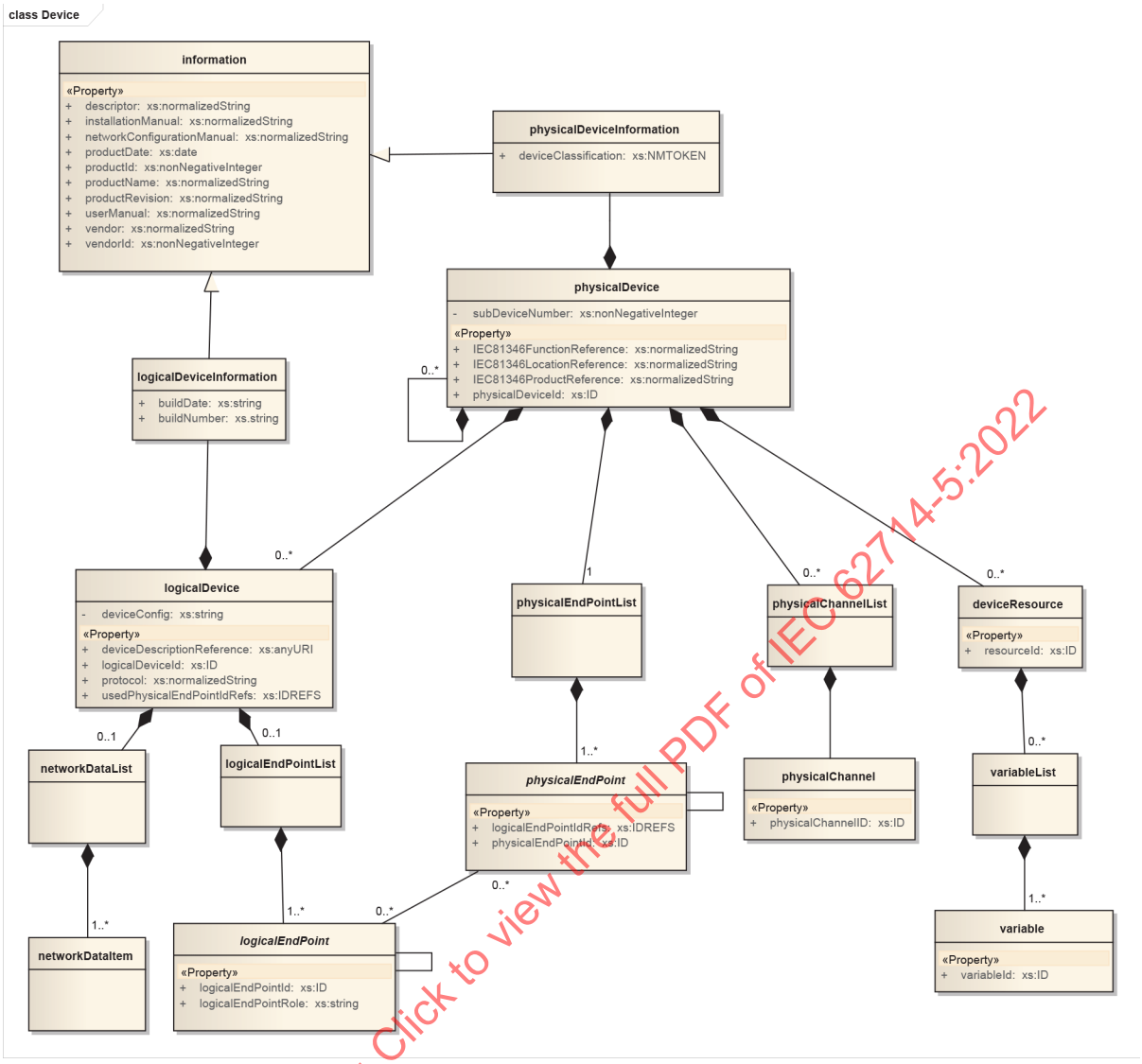


Figure 20 – Partie 1 du modèle de dispositif

NOTE L'élément logicalEndPoint est le point de transition vers la Figure 21 et y est répété.

La propriété physicalDeviceId est un GUID dans communicationStructure et doit identifier physicalDevice. S'il s'agit d'un sous-module à l'intérieur d'un dispositif, la propriété subDeviceNumber doit définir l'ordre spécifique au fabricant du sous-module dans le dispositif.

La propriété IEC81346FunctionReference doit décrire l'aspect fonctionnel du physicalDevice selon les règles de l'IEC 81346.

La propriété IEC81346LocationReference doit décrire l'emplacement du physicalDevice dans l'installation selon les règles de l'IEC 81346.

La propriété IEC81346ProductReference doit décrire les informations sur le produit du physicalDevice selon les règles de l'IEC 81346.

5.4.3 Élément Information

L'élément Information est un espace réservé pour tout type d'information associée au type de dispositif physique et logique. Il doit être utilisé pour décrire en détail les informations de type d'une instance de dispositif. Les propriétés proposées peuvent être améliorées sur demande.

5.4.4 Élément physicalDeviceInformation

L'élément physicalDeviceInformation développe le caractère générique Information et doit représenter les informations relatives au physicalDevice.

5.4.5 Élément logicalDeviceInformation

L'élément logicalDeviceInformation développe le caractère générique Information et doit représenter les informations relatives au logicalDevice.

5.4.6 Élément logicalDevice

L'élément logicalDevice doit représenter les aspects non physiques d'un dispositif réel. Souvent, les dispositifs de terrain plus complexes contiennent plusieurs logicalDevices avec leurs propres informations d'adresse et leurs tâches spécifiques.

logicalDevice doit regrouper les networkDataItems par l'intermédiaire de networkDataList, utilisée pour les tâches de configuration. Les logicalEndPoints doivent être également regroupés par l'intermédiaire de logicalEndpointlist. Outre la représentation de la topologie logique en 4.2, logicalEndPoint doit regrouper également les unités de données du protocole, échangées par l'intermédiaire de la connexion logique.

La propriété deviceConfig doit décrire les informations utilisées lors du démarrage du logicalDevice. Elles peuvent être échangées au moment de l'exécution entre différents partenaires de communication, en fonction du système de communication.

La propriété deviceDescriptionReference doit faire référence à un fichier de description de périphérique externe, s'il est disponible.

La propriété logicalDeviceId est un GUID dans le système d'automatisation et doit identifier le logicalDevice.

La propriété protocol doit décrire le protocole de communication utilisé, par exemple un bus industriel fondé sur Ethernet, un bus de terrain, etc.

La propriété usedPhysicalEndPointIdRefs doit représenter une liste de GUID avec les physicalEndPoints associés. Cela signifie que ces points d'extrémité sont utilisés par le logicalDevice.

5.4.7 Élément networkDataList

L'élément networkDataList doit représenter les networkDataItems associés au logicalDevice.

5.4.8 Élément networkDataItem

Un networkDataItem doit représenter un élément de configuration spécifique qui appartient au logicalDevice. Les networkDataItems types peuvent être l'adresse MAC, l'adresse IP, le masque de sous-réseau, le nom de la station, etc.

5.4.9 Élément logicalEndPointList

Cet élément est un modèle pour énumérer tous les logicalEndPoints disponibles.

5.4.10 Élément physicalEndPointList

Cet élément est un modèle pour énumérer tous les physicalEndPoints disponibles.

5.4.11 Élément physicalChannelList

L'élément physicalChannelList doit représenter un espace réservé pour les éléments de physicalChannel.

5.4.12 Élément physicalChannel

L'élément physicalChannel doit représenter un point de connexion à un signal, par exemple une entrée ou une sortie analogique ou binaire. Le type de données n'est pas explicitement modélisé ici. physicalChannel est répertorié par l'intermédiaire de physicalChannelList.

La propriété physicalChannelId est un GUID dans communicationStructure et doit identifier le canal ou le signal.

5.4.13 Élément deviceResource

L'élément deviceResource doit représenter un environnement d'exécution (par exemple, une unité centrale ou une machine virtuelle avec un logiciel spécifique) à l'intérieur d'un dispositif qui permet d'exécuter des programmes ou des parties de programmes avec ses fonctions d'interface homme-machine ou d'interface capteur et actionneur. L'élément est une représentation du terme RESOURCE utilisé dans l'IEC 61131-3.

5.4.14 Élément variableList

Cet élément est un modèle pour énumérer toutes les variables disponibles dans un deviceResource.

5.4.15 Élément variable

L'élément variable représente une variable, par exemple d'un programme de contrôleur ou dans un autre type de contrôleur. Le type de données n'est pas explicitement modélisé ici. L'élément variable doit représenter une VARIABLE à l'intérieur d'un composant programmable, par exemple un contrôleur selon la définition de l'IEC 61131-3. L'élément variable est déjà défini dans l'IEC 62714-4 comme une interface et des références à un élément variable de PLCopen XML (IEC 61131-10) au moyen du GUID.

La propriété variableId est un GUID dans communicationStructure et doit identifier la variable.

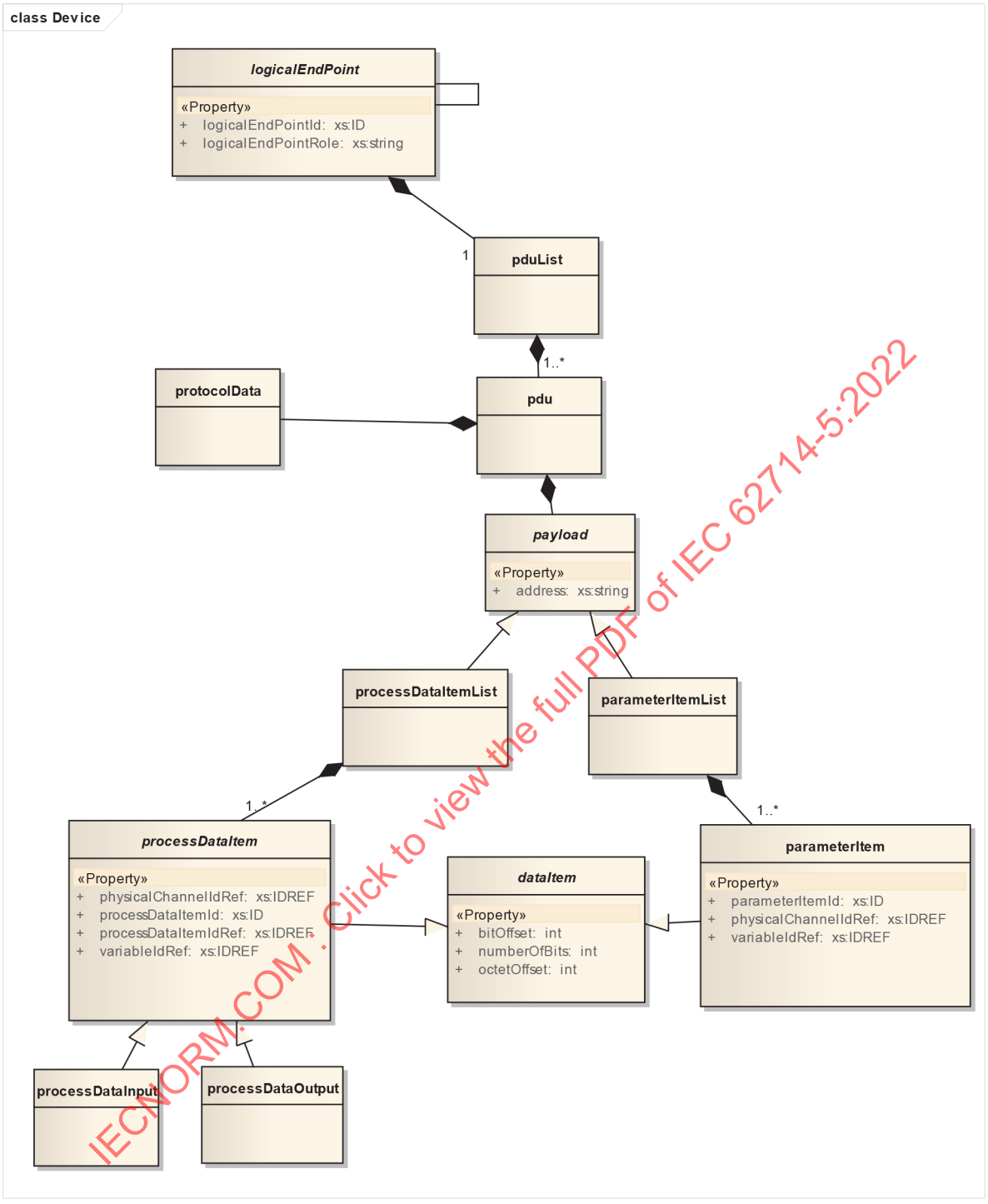


Figure 21 – Partie 2 du modèle de dispositif

NOTE L'élément logicalEndPoint est le point de transition de la Figure 20 et est répété ici.

5.4.16 Élément pduList

L'élément pduList doit représenter les unités de données de protocole échangées par l'intermédiaire du logicalEndPoint.

5.4.17 Élément pdu

L'élément pdu doit représenter une unité de données de protocole spécifique liée à cette logicalConnection. Il doit regrouper les protocolData et le payload (charge utile). La charge utile peut être spécialisée en processDataItemList ou parameterItemList.

5.4.18 Élément protocolData

L'élément protocolData doit être utilisé pour décrire la surcharge du protocole, qui n'est normalement pas modélisée par l'élément processDataItem ou parameterItem. Pour les protocoles de communication réels, des spécialisations de ces éléments peuvent être définies.

5.4.19 Élément payload

L'élément payload doit représenter le processDataItem et le parameterItem en tant qu'élément abstrait généralisé. L'adresse de propriété facultative peut être utilisée à des fins d'adresse ou d'identification d'un dataItem.

5.4.20 Élément processDataItemList

Cet élément est un modèle pour énumérer tous les processDataItems.

5.4.21 Élément ItemItemList

Cet élément est un modèle pour énumérer tous les parameterItems. En outre, un pdu de ce type doit possèder une adresse spécifique au protocole de communication.

5.4.22 Élément dataItem

L'élément dataItem doit représenter un objet de données communiqué par l'intermédiaire de la logicalConnection.

La propriété octetOffset doit définir le décalage d'octet de l'objet de données dans la charge utile du PDU. Le comptage doit commencer avec le premier octet de la charge utile en commençant par l'index '0' pour le premier octet à envoyer/recevoir.

La propriété bitOffset doit définir le décalage binaire de l'objet de données dans un octet adressé de la charge utile du PDU.

Les propriétés octetOffset et bitOffset peuvent être utilisées alternativement. Si les deux propriétés sont utilisées, elles doivent être additionnées afin de calculer la position du bit.

La propriété numberOfBits doit définir la longueur de l'objet de données en bits.

5.4.23 Élément processDataItem

L'élément processDataItem doit représenter un objet de données de processus communiqué par l'intermédiaire de la logicalConnection. Ces objets de données sont souvent transmis au moyen de primitives de service cycliques du système de communication.

La propriété processDataItemId est un GUID dans le système d'automatisation et doit identifier le processDataItem.

La propriété variableIdRef doit être la référence du processDataItem dans l'image de données de processus associée à une variable du contrôleur ou du programme de contrôleur. La référence est définie, si le physicalDevice est un contrôleur.

La propriété processDataItemIdRef doit être la référence deprocessDataItem dans l'image de données de processus associée à une variable du programme du dispositif. La référence est définie, si le physicalDevice n'est pas un contrôleur.

La propriété physicalChannelIdRef doit être la référence du processDataItem à un physicalChannel d'un dispositif.

5.4.24 Élément processDataInput

L'élément processDataInput est une spécialisation de processDataItem et doit représenter le flux de données du processus en tant qu'entrée du processus contrôlé.

5.4.25 Élément processDataOutput

L'élément processDataOutput est une spécialisation de processDataItem et doit représenter le flux de données du processus en tant que sortie du processus contrôlé.

5.4.26 Élément parameterItem

L'élément parameterItem doit représenter un objet paramètre communiqué au moyen de la logicalConnection. Ces objets de données sont souvent transmis au moyen de primitives de service cycliques du système de communication.

La propriété parameterItemId est un GUID dans le système d'automatisation et doit identifier le parameterItem.

La propriété variableItemIdRef doit être la référence à une variable du programme du contrôleur. La référence est définie, si le physicalDevice est un contrôleur.

La propriété physicalChannelIdRef doit être la référence du processDataItem à un physicalChannel d'un dispositif.

6 Représentation dans AutomationML

6.1 Aperçu de la mise en correspondance

6.1.1 Introduction de la mise en correspondance

La méthode de modélisation des systèmes de communication dans AutomationML repose sur l'utilisation de <RoleClass>es et de <InterfaceClass>es dédiées ainsi que sur un processus d'application spécial. Cette méthode reprend la distinction entre les réseaux d'applications et les réseaux physiques et leur interconnexion telle que décrite aux Articles 4 et 5.

6.1.2 Règles générales de mise en correspondance

Pour la représentation des différentes entités nommées aux Articles 4 et 5, les règles de mise en correspondance suivantes, telles qu'indiquées dans le Tableau 1, sont appliquées. La manière dont la sémantique d'un objet est représentée et celle dont une entité connexe est modélisée sont distinguées.

Tableau 1 – Règles de mise en correspondance

Entité d'information du modèle UML	Modélisation sémantique	Modélisation des entités
logicalTopology	RoleClass LogicalNetwork	InternalElement
logicalConnection	RoleClass LogicalConnection	InternalElement
logicalEndpoint	InterfaceClass LogicalEndPoint	Interface
physicalTopology	RoleClass PhysicalNetwork	InternalElement
physicalConnection	RoleClass PhysicalConnection	InternalElement
physicalEndpoint	InterfaceClass PhysicalEndPoint	Interface
physicalDevice	RoleClass PhysicalDevice	InternalElement
Variable	InterfaceClass VariableInterface tel que défini dans l'IEC 62714-4	Interface
physicalChannel	InterfaceClass SignalInterface tel que défini dans l'IEC 62714-1	Interface
logicalDevice	RoleClass LogicalDevice	InternalElement
pduList	Implicitement modélisé par l'hébergement de InternalElement	
pdu	RoleClass CommunicationPackage	InternalElement
protocolData	Attribut ou InterfaceClass DatagrammObject	Attribut ou Interface
payload	InterfaceClass DatagrammObject	Interface
processDataItemList	Implicitement modélisé par l'hébergement de InternalElement	
parameterItemList	Implicitement modélisé par l'hébergement de InternalElement	
dataItem	InterfaceClass DatagrammObject	Interface
processDataItem	InterfaceClass DatagrammObject	Interface
parameterItem	InterfaceClass DatagrammObject	Interface

6.1.3 Principes de base

La base de la méthode de modélisation est la définition d'une bibliothèque de classes de rôles AutomationML et d'une bibliothèque de classes d'interface AutomationML pour la modélisation des systèmes de communication et la dérivation à partir de celles-ci des classes de rôles et des classes d'interface pertinentes pour les cas d'application spéciaux suivant la définition de l'application des classes de rôles et des classes d'interface définie dans l'IEC 62714-1.

La bibliothèque de classes de rôles de communication AutomationML contient des rôles dédiés à l'identification d'éléments internes tels que les dispositifs physiques, les connexions physiques et les réseaux physiques, ainsi que des éléments internes tels que les dispositifs logiques, les connexions logiques et les réseaux logiques.

La bibliothèque de classes d'interface AutomationML contient des classes d'interface pour les points d'extrémité physiques et les points d'extrémité logiques.

La bibliothèque de classes de rôles de communication et la bibliothèque de classes d'interface de communication sont représentées à la Figure 22.