



<AutomationML/>

**The Glue for Seamless
Automation Engineering**

**Whitepaper AutomationML
Communication**

Document Identifier: WP Comm, V 1.0.0

State: September 2014

Table of content

Table of content	3
List of figures	6
List of tables	8
1 Introduction / Scope	10
2 Terms, definitions and abbreviations	11
2.1 Terms and definitions.....	11
2.1.1 AML attribute	11
2.1.2 AML document	11
2.1.3 AML class.....	11
2.1.4 AML file	11
2.1.5 AML interface	11
2.1.6 AML library	11
2.1.7 AML object	11
2.1.8 AutomationML / AML.....	11
2.1.9 Automation object.....	11
2.1.10 Instance	12
2.1.11 Instance hierarchy	12
2.1.12 Link	12
2.2 Abbreviations	12
2.3 Normative References	12
3 Use Cases and considered network structures	14
3.1 Use cases	14
3.1.1 UI031: Lossless transfer of communication device instance information	15
3.1.2 UI032: Lossless transfer of communication system information.....	17
3.2 Delimitation of modelling range	18
3.2.1 Considered interaction structures and life cycles.....	18
3.2.2 Considered network objects.....	19
3.2.3 Considered network structures	20
3.2.4 Considered communication content.....	25
3.3 Derived modelling requirements	25
3.4 Possible advances	26
4 UML model	27
4.1 Overview	27
4.2 Logical topology	27
4.2.1 Item logicalTopology	28
4.2.2 Item logicalConnection	28
4.2.3 Item logicalEndPoint	28
4.3 Physical topology	28
4.3.1 Item physicalTopology	29
4.3.2 Item physicalConnection	29
4.3.3 Item physicalEndPoint.....	29
4.4 Device	29
4.4.1 Item physicalDevice	29

4.4.2	Item Information	30
4.4.3	Item variable	31
4.4.4	Item physicalChannel	31
4.4.5	Item logicalDevice	31
4.4.6	Item pduList	31
4.4.7	Item pdu	31
4.4.8	Item protocolData	31
4.4.9	Item payload	31
4.4.10	Item processDataItemList	31
4.4.11	Item parameterItemList	31
4.4.12	Item dataItem	32
4.4.13	Item processDataItem	32
4.4.14	Item processDataInput	34
4.4.15	Item processDataOutput	34
4.4.16	Item parameterItem	34
5	Representation within AutomationML	35
5.1	Overview of mapping	35
5.1.1	General mapping rules	35
5.1.2	Basics	35
5.1.3	Application process	37
5.2	Basic communication role class library	39
5.2.1	General	39
5.2.2	RoleClass PhysicalDevice	40
5.2.3	RoleClass PhysicalEndpointlist	40
5.2.4	RoleClass PhysicalConnection	40
5.2.5	RoleClass PhysicalNetwork	41
5.2.6	RoleClass LogicalDevice	41
5.2.7	RoleClass LogicalEndpointlist	41
5.2.8	RoleClass LogicalConnection	42
5.2.9	RoleClass LogicalNetwork	42
5.3	Basic communication interface class library	42
5.3.1	General	42
5.3.2	InterfaceClass PhysicalEndPoint	43
5.3.3	InterfaceClass LogicalEndPoint	43
5.4	Steps to model technology specific libraries	43
5.4.1	General	43
5.4.2	Step 1: Development of technology specific role classes	43
5.4.3	Step 2: Development of technology specific interface classes	44
5.4.4	Step 3: Development of system unit class libraries	44
5.4.5	Step 4: Modelling the network	45
5.4.6	Step 5: Modelling the connections	45
5.5	PDU modelling	46
5.5.1	RoleClass CommunicationPackage	46
5.5.2	InterfaceClass DatagramObject	48
5.5.3	Steps to model technology specific libraries	49
5.6	References to attributes	51

5.7	Usage of Metadata.....	54
6	Examples.....	55
6.1	Lab size manufacturing system model	55
6.1.1	Role classes	56
6.1.2	Interface classes	57
6.1.3	System Unit Classes	58
6.1.4	Instance Hierarchy	65
6.1.5	Links	67
6.2	PROFINET demonstrator.....	68
6.2.1	Role classes	69
6.2.2	Interface classes	71
6.2.3	System Unit Classes	72
6.2.4	Instance Hierarchy	78
6.2.5	Links	81
7	Bibliography	82

List of figures

Figure 1 - General Engineering activities communication system engineering is embedded within	14
Figure 2 - Execution sequence of the use case UI031	15
Figure 3 - Alternative execution sequence of the use case UI031	16
Figure 4 - Execution sequence of the use case UI 032	17
Figure 5 - Example of a logical level view on communication systems	19
Figure 6 - Example of a physical level view on communication systems	20
Figure 7 - Combined views on communication systems	20
Figure 8 - Star topology example	21
Figure 9 - Ring topology example	21
Figure 10 - Line topology example	22
Figure 11 - Simple network with direct wiring	22
Figure 12 - Network with active infrastructure	23
Figure 13 - Networks connected by gateways	23
Figure 14 - Hierarchical structured networks	24
Figure 15 - Network covering multiple applications	24
Figure 16 - General modelling strategy for PDUs	25
Figure 17 - Structure of communication	27
Figure 18 - View on logical topology	28
Figure 19 - View on physical topology	29
Figure 20 - Part 1 of the device model	30
Figure 21 - Part 2 of the device model	33
Figure 22 - Communication role class library and communication interface class library	36
Figure 23 - Derived role class libraries and interface class libraries for a special example	36
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 - Lab size manufacturing system model	55
Figure 47 - Physical and logical topology of network example	55
Figure 48 - Role class library for plant model example	56
Figure 49 - Interface class library for plant model example	57
Figure 50 - System unit class library for plant model example	58
Figure 51 - System unit class for physical device example 1	59
Figure 52 - System unit class for physical device example 2	60
Figure 53 - System unit class for physical device example 3	62
Figure 54 - System unit class for physical connection example	63
Figure 55 - System unit class for logical device example 1	64
Figure 56 - System unit class for logical device example 2	64
Figure 57 - System unit class for logical connection example	65
Figure 58 - Instance hierarchy for communication system example	67
Figure 59 - Internal links related to the instance hierarchy for communication system example.....	68
Figure 60 - Physical and logical topology of PROFINET demonstrator	68
Figure 61 - Communication packages of PROFINET demonstrator	69
Figure 62 - Role class library for PROFINET demonstrator example.....	70
Figure 63 - Interface class library for PROFINET demonstrator example	71
Figure 64 - System unit class library for PROFINET demonstrator example	72
<i>Figure 65 - System unit class for physical PROFINET device example 1.....</i>	<i>72</i>
<i>Figure 66 - System unit class for physical PROFINET device example 2.....</i>	<i>73</i>
Figure 67 - System unit class for physical PROFINET device example 3.....	74
<i>Figure 68 - System unit class for physical PROFINET connection example.....</i>	<i>75</i>
<i>Figure 69 - System unit class for logical PROFINET device example 1.....</i>	<i>75</i>
<i>Figure 70 - System unit class for logical PROFINET device example 2.....</i>	<i>76</i>
<i>Figure 71 - System unit class for logical PROFINET device example 3.....</i>	<i>77</i>
<i>Figure 72 - System unit class for logical PROFINET connection example</i>	<i>77</i>
<i>Figure 73 - System unit class for PROFINET communication package example</i>	<i>78</i>
Figure 74 - Instance hierarchy for PROFINET demonstrator example.....	80
Figure 75 - Internal links related to the instance hierarchy for PROFINET demonstrator example.....	81

List of tables

Table 1 - Abbreviations	12
Table 2: Mapping rules	35
Table 3 - RoleClass PhysicalDevice	40
Table 4 – RoleClass PhysicalEndpointlist	40
Table 5 - RoleClass PhysicalConnection	40
Table 6 - RoleClass PhysicalNetwork	41
Table 7 - RoleClass LogicalDevice	41
Table 8 - RoleClass LogicalEndpointlist	41
Table 9 - RoleClass LogicalConnection	42
Table 10 - RoleClass LogicalNetwork	42
Table 11 - InterfaceClass PhysicalEndPoint	43
Table 12 - InterfaceClass LogicalEndPoint	43
Table 13 - RoleClass CommunicationPackage	47
Table 14 - InterfaceClass DatagramObject	49
Table 15 - Communication related attributes	51
Table 16 - Field SourceDocumentatInformation according to communication related libraries	54
Table 17 - Role PhysicalEthernetConnection	56
Table 18 - Role PhysicalEthernetDevice	56
Table 19 - Role PhysicalEthernetEndpointlist	56
Table 20 - Role PhysicalEthernetNetwork	56
Table 21 - Role LogicalEthernetConnection	57
Table 22 - Role LogicalEthernetDevice	57
Table 23 - Role LogicalEthernetEndpointlist	57
Table 24 - Role LogicalEthernetNetwork	57
Table 25 - InterfaceClass EthernetPlug	57
Table 26 - InterfaceClass EthernetSocket	58
Table 27 - InterfaceClass LogicalEthernetEndPoint	58
Table 28 - SystemUnitClass PHOENIX FL IL 24 BK	59
Table 29 - SystemUnitClass WAGO 750 342	60
Table 30 - SystemUnitClass FL Switch 5TX	62
Table 31 - SystemUnitClass Ethernet Wire	63
Table 32 - SystemUnitClass Logical PHOENIX FL IL 24 BK	64
Table 33 - SystemUnitClass Logical WAGO 750 342	64
Table 34 - SystemUnitClass LogicalEthernetConnection	65
Table 35 - Role ProfinetExamplePhysicalConnectionRoleClass	70
Table 36 - Role ProfinetExamplePhysicalDeviceRoleClass	70
Table 37 - Role ProfinetExamplePhysicalEndpointlistRoleClass	70
Table 38 - Role ProfinetExamplePhysicalNetworkRoleClass	70
Table 39 - Role ProfinetExampleLogicalConnectionRoleClass	70
Table 40 - Role ProfinetExampleLogicalDeviceRoleClass	70
Table 41 - Role ProfinetExampleLogicalEndpointlistRoleClass	71
Table 42 - Role ProfinetExampleLogicalNetworkRoleClass	71
Table 43 - Role ProfinetPackage	71

Table 44 - InterfaceClass ProfinetExamplePhysicalPlug	71
Table 45 - InterfaceClass ProfinetExamplePhysicalSocket.....	71
Table 46 - InterfaceClass ProfinetExampleLogicalEndPointInterfaceClass.....	72
Table 47 - InterfaceClass ProfinetProfinetDatagrammObject	72
Table 48 - SystemUnitClass RFC 470 PN 3TX	73
Table 49 - SystemUnitClass PN BK DI8.....	73
Table 50 - SystemUnitClass AXL BK PN-ME.....	74
Table 51 - SystemUnitClass Ethernet Wire	75
Table 52 - SystemUnitClass main control application	75
Table 53 - SystemUnitClass Input function.....	76
Table 54 - SystemUnitClass Output function.....	77
Table 55 - SystemUnitClass LogicalProfinetConnection	77
Table 56 - SystemUnitClass ProfinetPackage.....	78

1 Introduction / Scope

Engineering processes of technical systems and its embedded automation systems have to be executed with increasing efficiency and quality. Especially the project duration has to be increased while the complexity of the engineered system increases. To solve this problem, the engineering process is more and more 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 forthcoming application of fieldbus and Ethernet technology they connect automation devices and further interacting entities and have to fulfil special requirements on communication quality, safety and security. Thus, within the engineering process of modern technical systems also communication system related engineering information and artefacts have 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 of communication components of automation devices including communication addresses and communication package structuring within PLC 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.

But the consistent and lossless transfer of communication system engineering information along the complete engineering chain of technical systems is unsolved up to now. While user organisations and companies have provided data exchange formats for parts of the relevant information like FCDML, EDDL, and GSDML the above named application cases cannot be covered by a data exchange format. Especially, the networking related information describing communication relations and its properties and quality cannot be modelled by a data exchange format.

This whitepaper will close this gap. It will provide a neutral, vendor and communication technology independent, and XML based methodology for communication system information exchange among engineering tools developed by AutomationML e.V.

2 Terms, definitions and abbreviations

2.1 Terms and definitions

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

2.1.1 AML attribute

property which belongs to an AML object

Note: AML attributes are described as XML element corresponding to IEC 62424:2008.

2.1.2 AML document

certain CAEX document following IEC 62714 including all referenced sub documents

Note: AML documents may be stored as files, but also e.g. as string, or data streams.

2.1.3 AML class

predefined AML object type

Note: AML classes are stored within AML libraries.

Note: AML classes define reusable sample solutions, characterized by attributes, interfaces, or aggregated objects.

Note: AML classes can be used for multiple instantiations.

2.1.4 AML file

certain CAEX file following IEC 62714 with the extension .aml excluding all referenced sub files

2.1.5 AML interface

single connection point that belongs to an AML object and can be linked with another interface

Note: Interfaces allow the description of relations between objects by the definition of CAEX InternalLinks. Examples are a signal interface, a product interface or a power interface.

2.1.6 AML library

library containing AML classes

2.1.7 AML object

data representation of an automation object or a group of automation objects

Note: The AML object is the core element of AML. It may contain administration items, attributes, interfaces, relations, or references. An AML object is an individual instance and derived from standard AML classes.

Note: AML objects have a relation to their corresponding AML class.

Note: Examples for relations are type-instance-relations and copy-instance-relations. Single instances can be extended, e.g. by aggregated objects or attributes.

Note: AML objects have a copy-instance-relation to their AML class.

2.1.8 AutomationML / AML

XML based data exchange format for plant engineering data

2.1.9 Automation object

entity in an automated system

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

2.1.10 Instance

data representation of a concrete real world item or concrete logical engineering item

2.1.11 Instance hierarchy

hierarchy of AML objects

2.1.12 Link

connection between objects of the top-level format CAEX

Note: A link is modelled by means of CAEX InternalLink.

2.2 Abbreviations

For the purpose of this document the abbreviations listed in Table 1 apply.

Table 1 - Abbreviations

Abbreviation	Meaning
AML	Automation Markup Language
AutomationML	Automation Markup Language
CAEX	Computer Aided Engineering Exchange
ECAD	Computer aided engineering for electrical engineering
EDD	Electronic Device Description
EDS	Electronic Data Sheet
FDCML	Field Device Configuration Markup Language
GUID	Global Unique Identifier
GSDML	Generic Station Description Markup Language
GSD	General Station Description
ID	Identifier
MCAD	Computer aided engineering for mechanical engineering
PDU	Protocol Data Unit
UML	Unified Modelling Language
UUID	Universal Unique Identifier
XML	Extensible Markup Language

2.3 Normative References

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

IEC 60027 (all parts), Letter symbols to be used in electrical technology

IEC 60050, International Electrotechnical Vocabulary

IEC 62424:2008, Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools

ISO/IEC 9834-8, Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components

IEC 62714 (all parts), Engineering data exchange format for use in industrial systems engineering – AutomationML

ISO 80000-1, Quantities and units – Part 1: General

Extensible Markup Language (XML) 1.0:2004, W3C Recommendation (available at <<http://www.w3.org/TR/2004/REC-xml-20040204/>>)

IEC 61158, Industrial communication networks – Fieldbus specifications

IEC 61784 Industrial communication networks – Profiles

IEC 61346 Industrial systems, Installations and Equipment and Industrial Products — Structuring Principles and Reference Designations

3 Use Cases and considered network structures

The modelling of communication systems based on AutomationML targets the modelling of a large scale 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 section the use cases for the application of the communication system modelling as well as the relevant information sets within them are named.

3.1 Use cases

Communications system related information is relevant in various engineering activities along the engineering chain of production systems. Within the engineering process of production systems communication system can be designed in the detailed engineering phase exploiting various tools. Thereby, communication system related information is created which subsequently should 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.

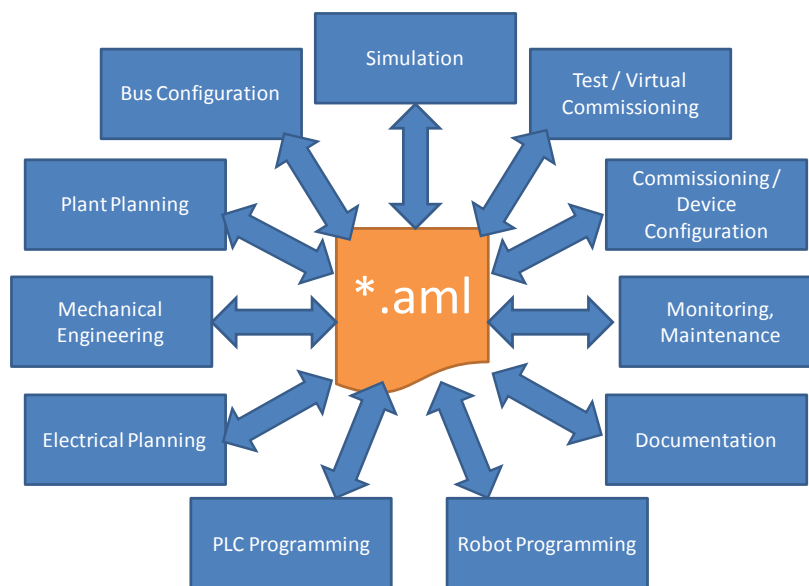


Figure 1 - General Engineering activities communication system engineering is embedded within

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

- Plant planning tools
- Mechanical engineering tools (MCAD)
- Electrical engineering tools (ECAD)
- PLC 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 configuration 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 should be exchanged between engineering tools. These two use cases are the main target of the modelling of communication systems based on AutomationML.

3.1.1 UI031: 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 PLC 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).

Within the related engineering activities engineers with the engineering roles of PLC programmer, HMI programmer, electrical design engineer, commissioner communication, commissioner PLC, and robot programmer can be involved. They will execute the following sequence of engineering and data exchange activities which has to 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 PLC programming and device configuration tools
- Step 4.** Integration of device descriptions (like GSDML) in PLC programming tool
- Step 5.** Design of PLC programs and configurations within PLC programming tool
- Step 6.** Export of communication device relevant information from PLC 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)

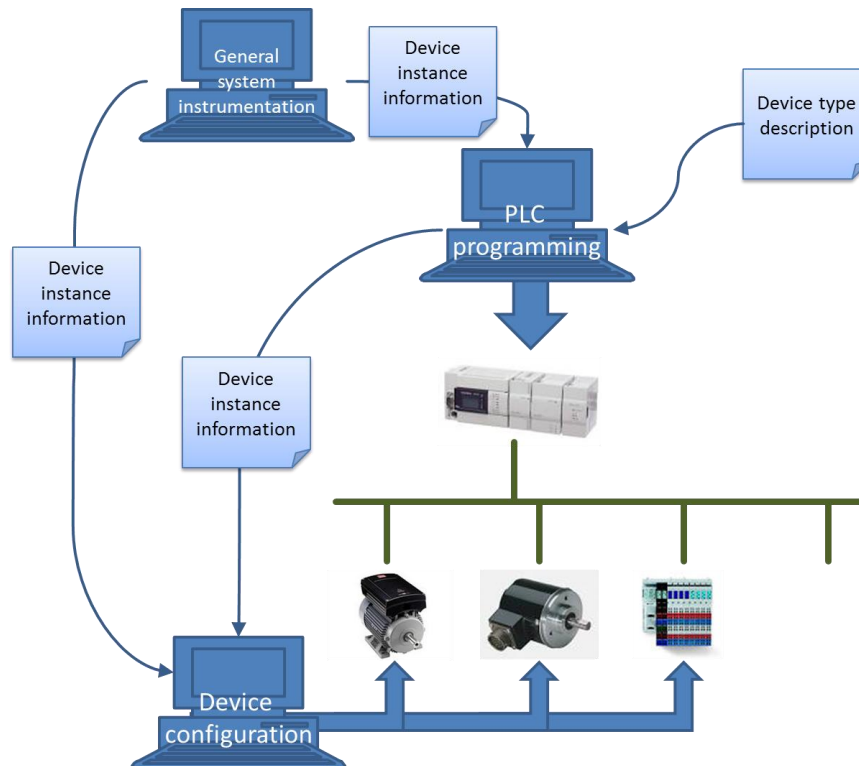


Figure 2 - Execution sequence of the use case UI031

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

- 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 parameterisation of communication component of device (addresses, etc.) and for structuring of communication packages (send data points)

- 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 PLC programming tool
- Step 6.** Use of information from peripheral devices inside the PLC program

Both sequences are commonly depicted in Figure 3.

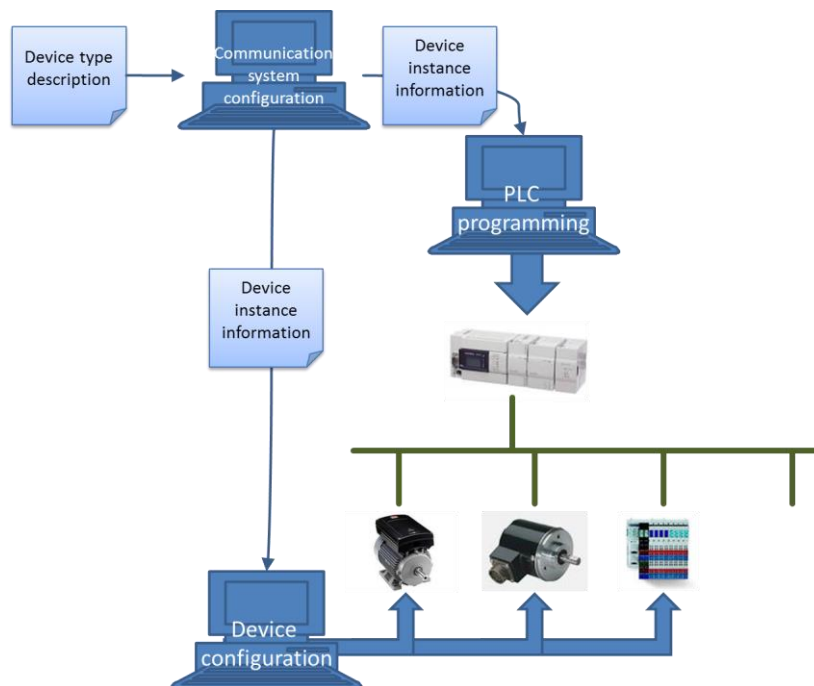


Figure 3 - Alternative execution sequence of the use case UI031

Possible tools exporting communication system information can be e.g. PLC programming tools. They cover PLC 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 has to cover information about IO lists, association of variables (I/Os) to communication data packages and device parameters for parameterisation of communication devices like address like IP address, media

access like MAC address, subnet masks, gateway addresses, and used communication objects like profile information.

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

3.1.2 UI032: Lossless transfer of communication system information

Beyond the configuration of communication devices communication system information are 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 shall be provided to device configuration tools, documentation and maintenance tools, and network management tools. They shall enable the combination of physical wiring with logical communication connections for error detection.

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

- Step 1.** Engineering of sets of variables to be transmitted within communication system within PLC programming tools and sensor/actuator programming/configuration tools
- Step 2.** Export of these variable sets from PLC 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

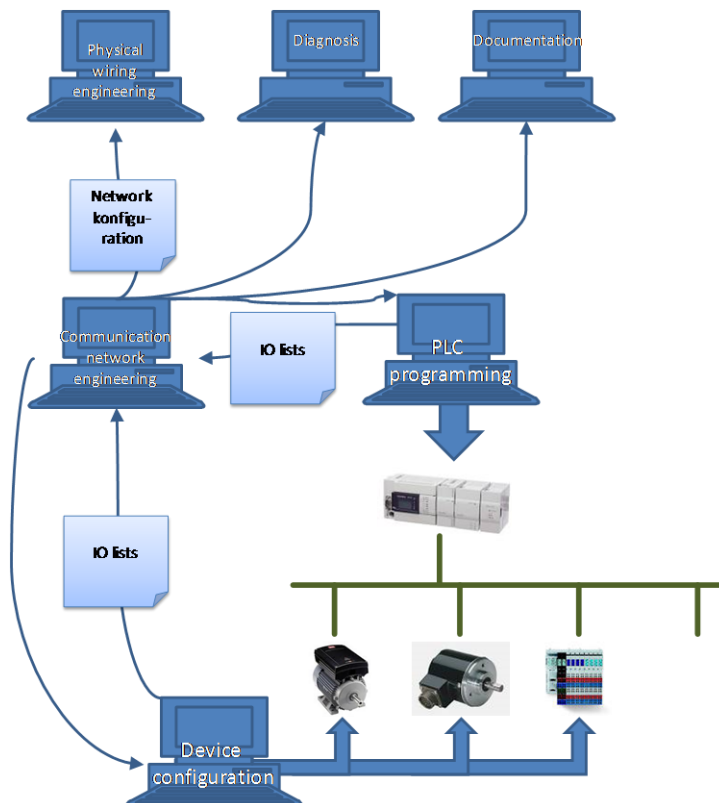


Figure 4 - Execution sequence of the use case UI 032

Source of the engineering data exchanged can be PLC programming tools covering PLC 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 has to cover information about network topology, connector lists of devices and its properties, physical connections between connectors of devices and its properties, logical links between device based applications and its 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, used protocols, etc.

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

3.2 Delimitation of 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.

3.2.1 Considered 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 on 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 its describing 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 covered usually by device description languages.

Nevertheless, the use of AutomationML to model communication system with these properties is not excluded but also not intended.

Communication system descriptions can cover different levels of detail. They can range from very abstract descriptions only naming communicating partners and its dependencies down to a detailed description of communication connections and its properties as well as device and application parts with its 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 PLC based control systems can serve. They usually embed different fieldbus systems, Ethernet based communication, copper and fibre optic wiring, wireless communication, different active infrastructure, etc. On one wire different communication protocols can run. On devices several control application parts are executed. 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 like 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 its impact on model content and model use should be covered by the AutomationML based modelling of communication systems.

3.2.2 Considered network objects

Within the following the considered network structures and information sets describing them should be detailed.

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] 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) have to 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.

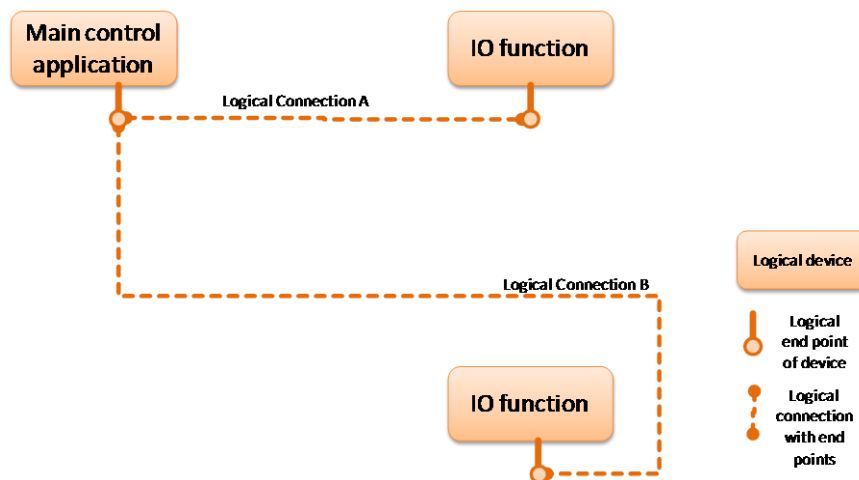


Figure 5 - Example of a logical level view on communication systems

The logical network may contain different objects with different describing properties. While logical devices control 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 describing properties can be considered as attributes of the objects of interest.

Looking on the physical level physical devices with physical interfaces can be found. In the example given in Figure 6 a PLC 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 describing properties. While physical devices may have processor capacity or identifiers, physical end points may have an address or a maximal data rate, and physical connections may have a possible transmission rate, a wire type, or a documentation number. In any case these describing properties can be considered as attributes of the objects of interest again.

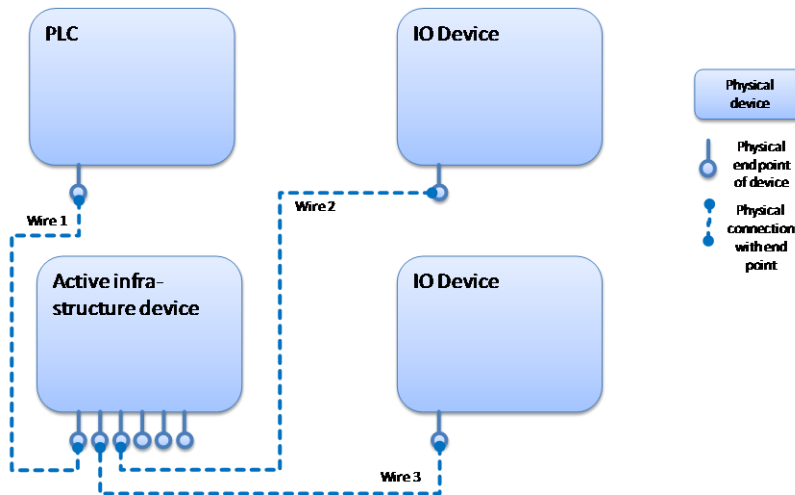


Figure 6 - Example of a physical level view on communication systems

Both views need to be combined to get the complete network description. Therefore, logical devices are hosted by physical devices. In the example given in Figure 7 a PLC hosts a main control application while the IO devices host IO functions. In addition, physical interfaces and logical interfaces are mapped. Thereby 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 like it is not given in some communication technologies.

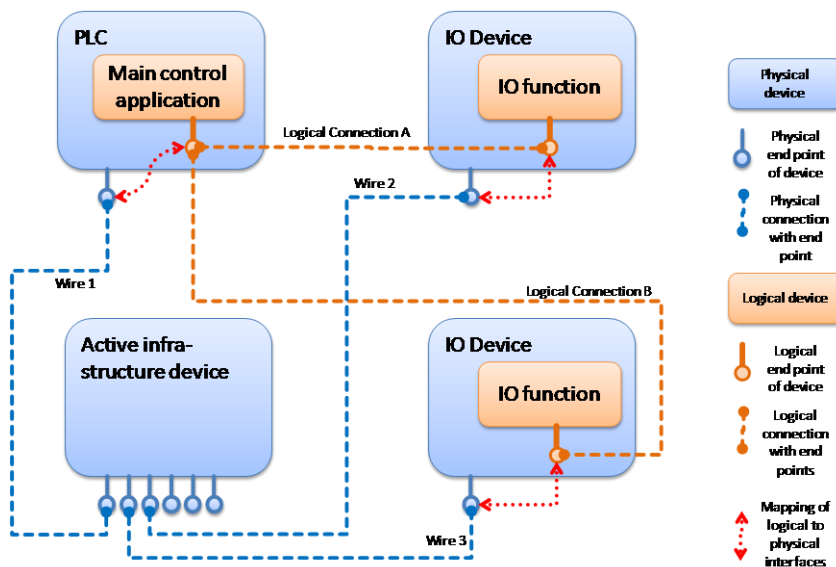


Figure 7 - Combined views on communication systems

3.2.3 Considered network structures

Within industrial communication systems different network architectures can be considered. On the one hand different wiring topologies can be distinguished. On the other hand different network

integration structures are possible. The communication system modelling presented here should be applicable to all of them.

With respect to different wiring topologies the intended modelling should cover at least star, line, and ring topologies including its combinations. Some examples are given in Figure 8, Figure 9, and Figure 10.

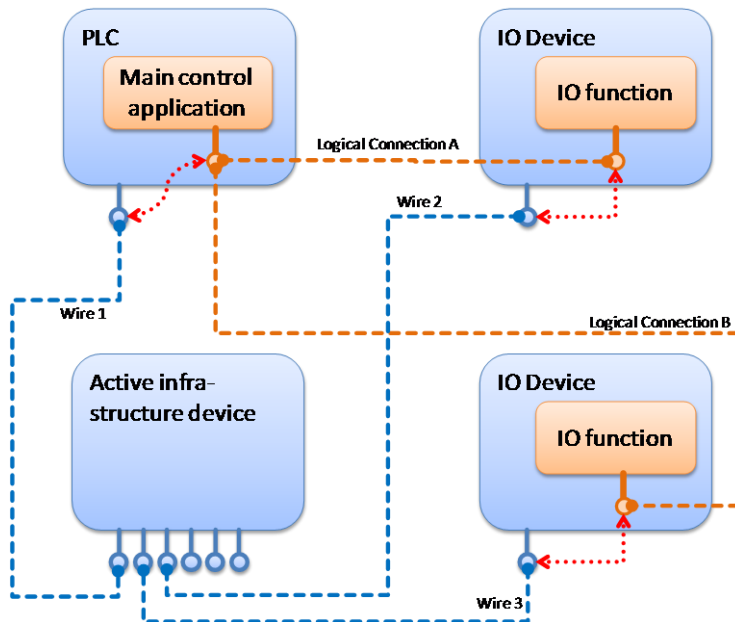


Figure 8 - Star topology example

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

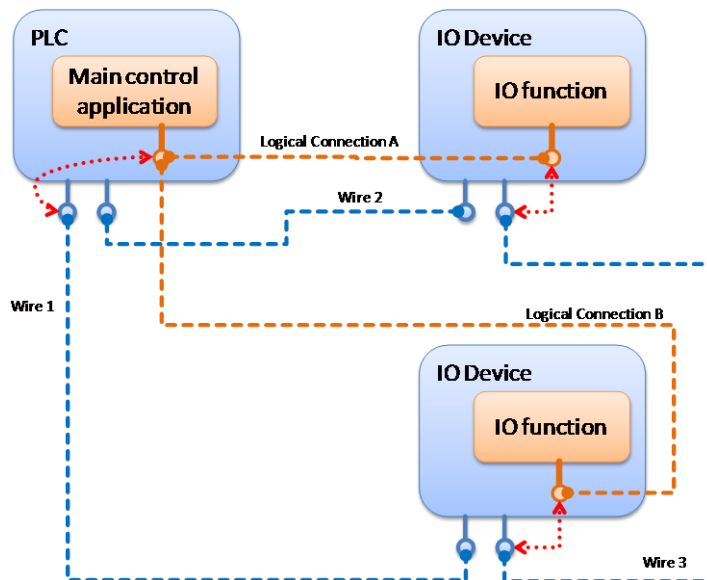


Figure 9 - Ring topology example

Figure 9 presents a standard 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.

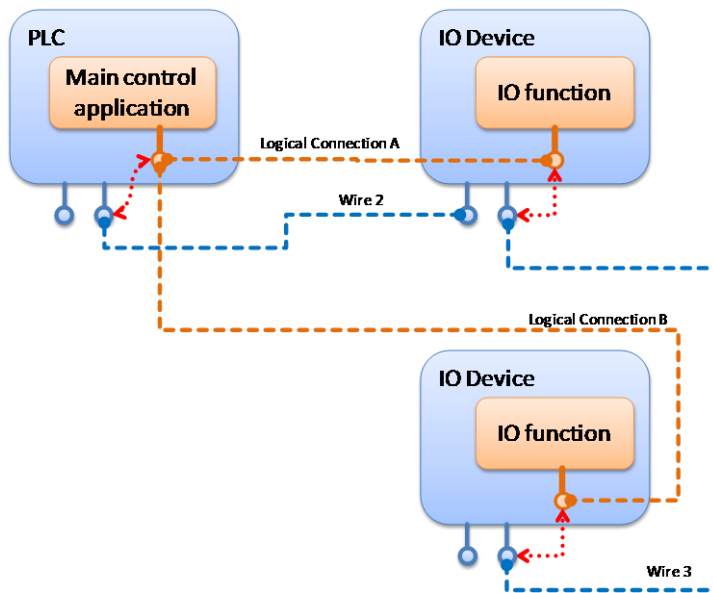


Figure 10 - Line topology example

Figure 10 presents a standard 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 integration 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.

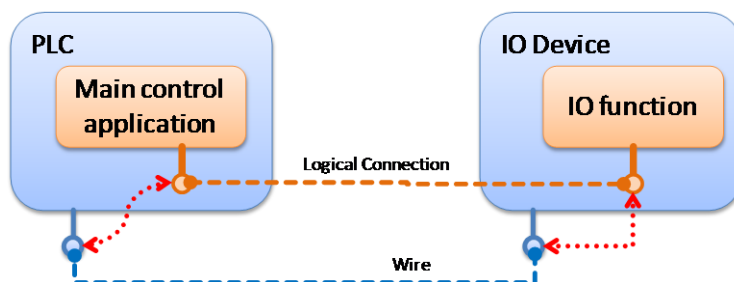


Figure 11 - Simple network with direct wiring

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 has to be presentable.

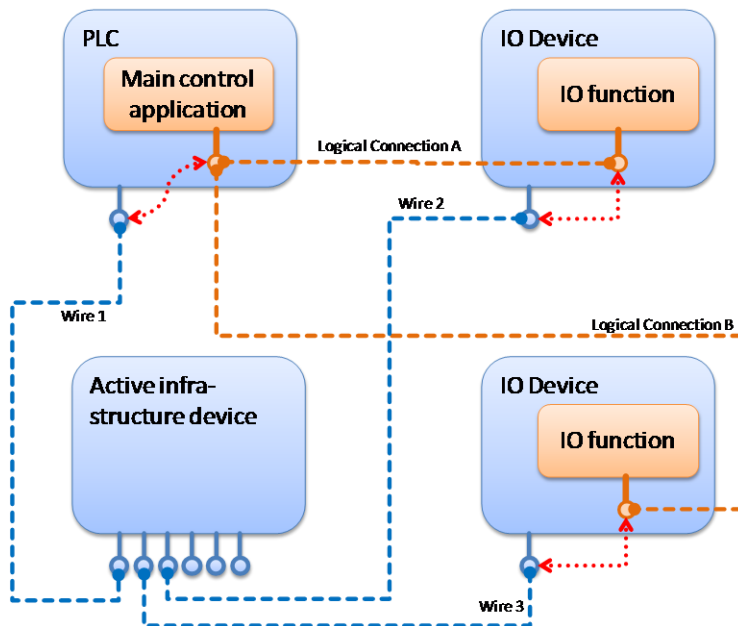


Figure 12 - Network with active infrastructure

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

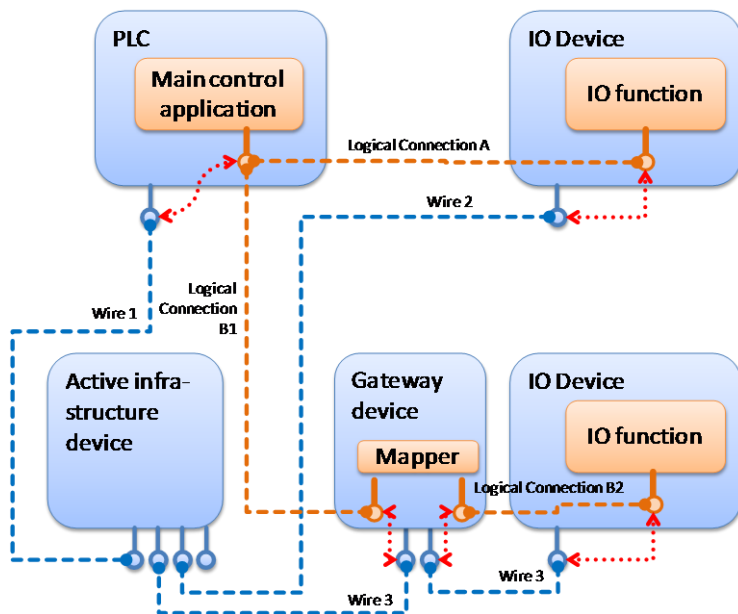


Figure 13 - Networks connected by gateways

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.

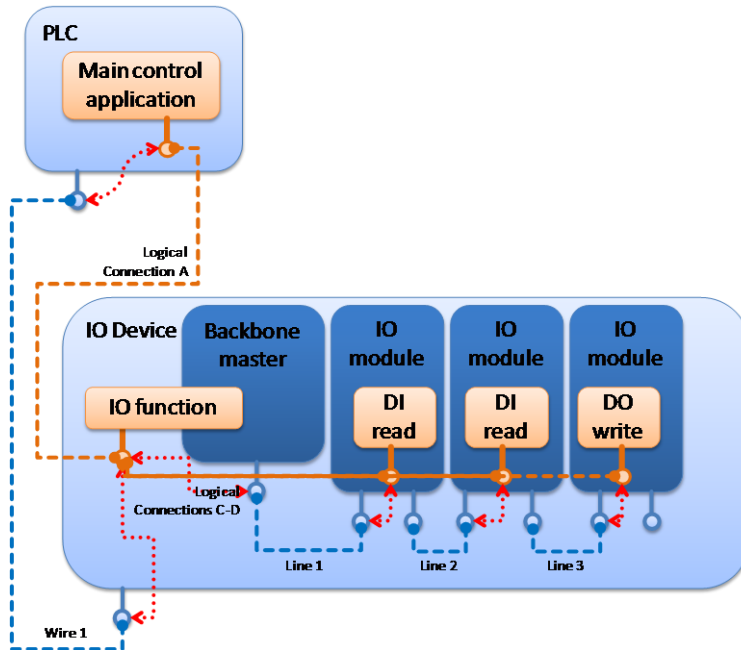


Figure 14 - Hierarchical structured networks

Figure 14 represents a hierarchical network structure as usually given in the case of a communication system containing a physical device with a backbone 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.

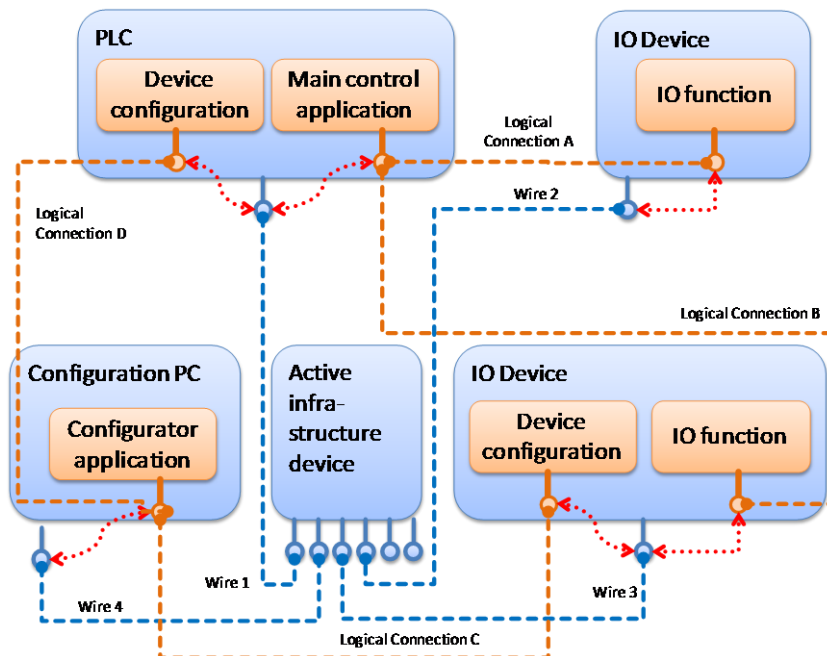


Figure 15 - Network covering multiple applications

3.2.4 Considered communication content

Within communication systems communication datagrams (known as Protocol Data Units / PDU) are exchanged between control application parts. Hence, they belong to a logical connection.

Each PDU contains control information (sensor and actor signals, status, alarms, etc.) modelled in AutomationML based on interfaces of PLCopenXMLInterface type (see AutomationML Whitepaper Part 4 - Logic Description).

Thus, each logical connection has to contain PDU objects exchanged via this connection. Each of the PDU objects is linked to a PLCopenXMLInterface or a SignalInterface modelling the exchanged information.

This structure is depicted in Figure 16

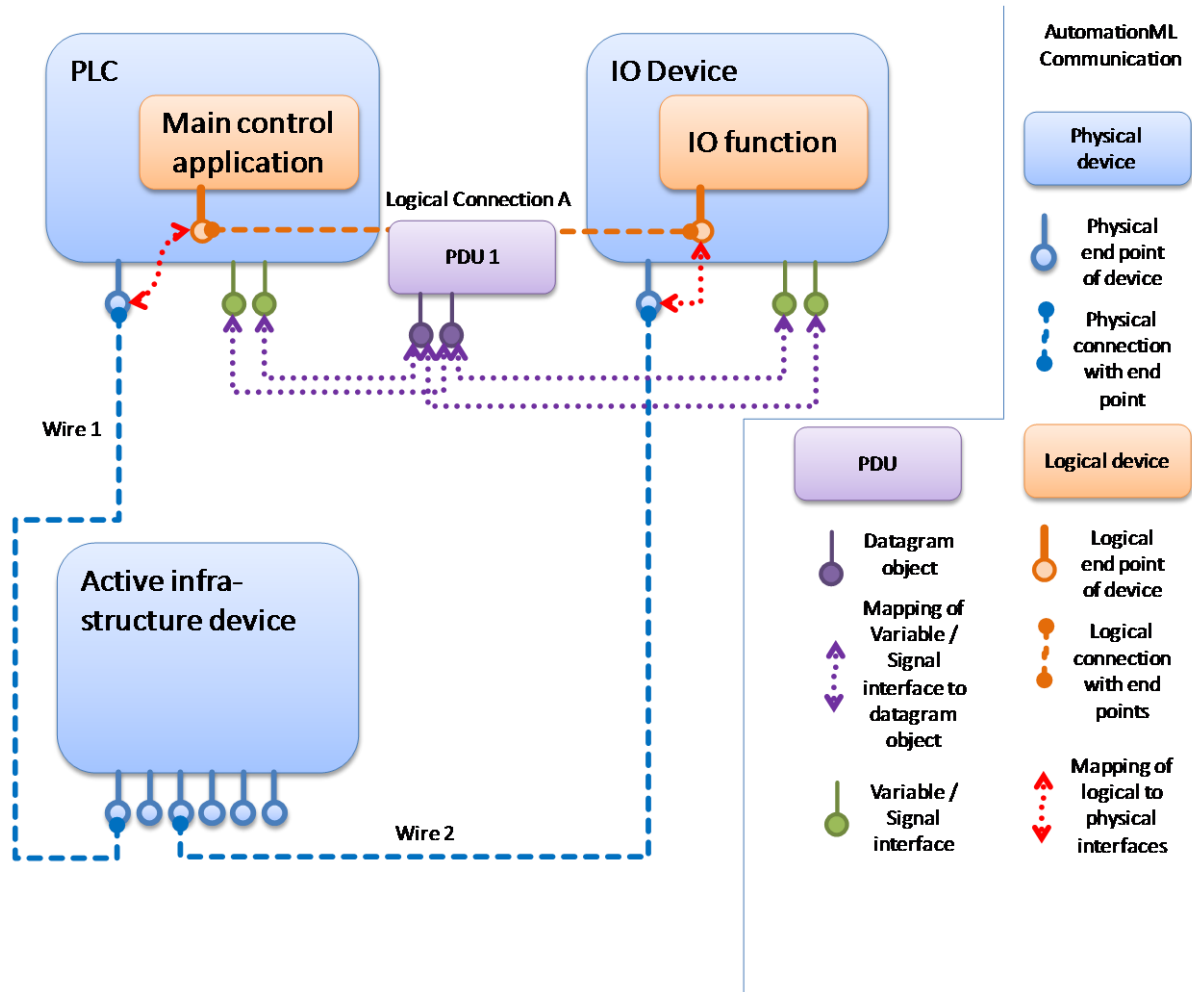


Figure 16 - General modelling strategy for PDUs

3.3 Derived modelling requirements

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

1. Logical level information
 - a. Logical devices representing application parts with its describing properties
 - b. End points of logical connections attached to logical devices representing the application part interface with its describing properties
 - c. Logical connections representing the information exchange between application parts with its describing properties
 - d. Process data units modelling the exchanged data packages.

2. Physical level information
 - a. Physical devices representing the physically communicating entities with its properties
 - b. End points of physical connections attached to physical devices representing the physical interface of the devices (socket) with its describing properties
 - c. Physical connections representing the physical wiring or used wireless channels with its describing properties
3. Relations between logical and physical levels
 - a. Mapping of end points to represent the implementation of logical connections by physical connections
 - b. Relations between variables exchanged and applied communication technology / protocol especially by mapping of variables to communication package structures
4. Network systems
 - a. Unique association of devices and connections to networks
 - b. Unique hierarchical network structuring
 - c. Dependencies between networks
 - d. Dependencies between devices like redundancy properties
 - e. Dependencies between connections like redundancy properties

3.4 Possible advances

The modelling of communication systems by AutomationML mainly targets existing control applications as well as existing communication technologies. But against the background of the Industry 4.0 initiative in Germany [3] additional applications can be envisioned even if they are not described in detail in this whitepaper.

Industry 4.0 envisions the on-demand design of interaction structures of system entities. Especially there will be manufacturing components implemented by cyber physical production systems (CPPS) interacting with each other and with product controlling entities and cloud based services. These on-demand interactions require an appropriate representation of services provided by CPPS and cloud based services including its technical access paths and semantics. Such descriptions can be based on the description of logical and physical devices named above which can be automatically applied for interaction establishing.

In addition the modelling of complete communication systems can be exploited in the Industry 4.0 context to log applied connections and to store such logs for production process control documentation purposes.

Within the following representation of communication system modelling capabilities this future application possibilities are basically considered.

4 UML model

In the following a more detailed description of communication system related objects and properties and its describing information modelled by AutomationML is given.

4.1 Overview

The UML model uses class diagrams showing the static relationships between entities. The entities are described here always as types. The real implementation of this model depends on tool preferences. E.g. the aggregation demonstrates here the "has a" association or relationship between the instances of 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 we are not able to define it explicitly.

Figure 17 explains the focal points of the modelled communication structure of one 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.

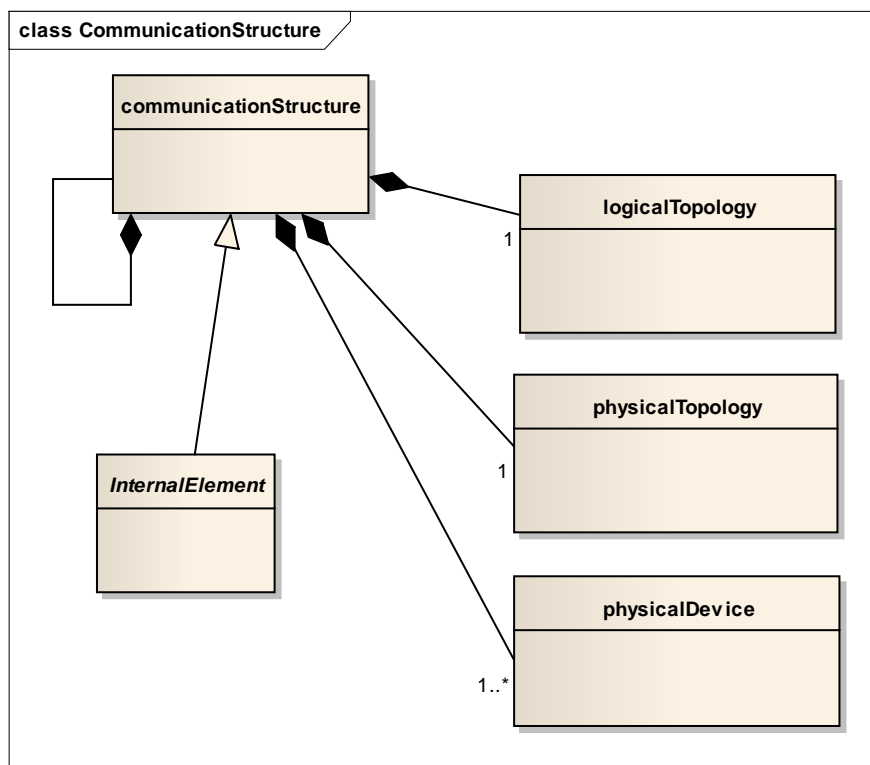


Figure 17 - Structure of communication

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

4.2 Logical topology

The logical topology describes the logical view on 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

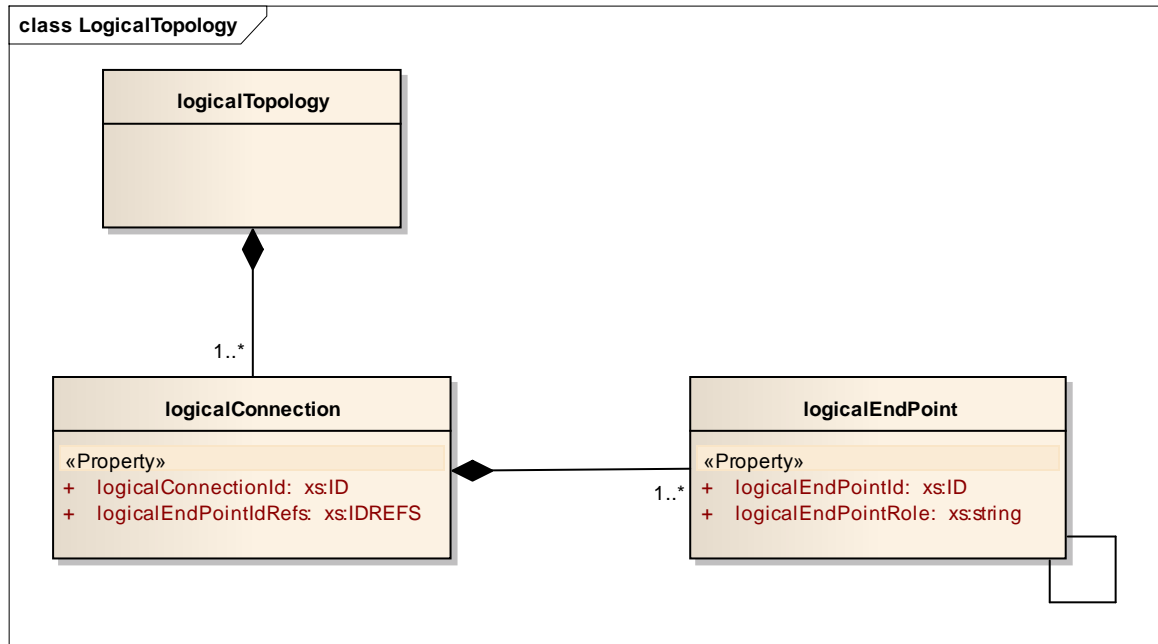


Figure 18 - View on logical topology

4.2.1 Item logicalTopology

The class *logicalTopology* is the entry point for modelling the logical topology of an automation network. It aggregates all communication relations between logical end points, associated to automation devices.

A real derivation of *logicalTopology* may be the representation of logical networks in a plant.

4.2.2 Item logicalConnection

The entity *logicalConnection* describes the communication relation between two logical end points.

The property *logicalConnectionId* is a GUID inside the plant and identifies the *logicalConnection*.

The property *logicalEndPointIdRefs* is an array containing 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.

4.2.3 Item logicalEndPoint

The entity *logicalEndPoint* represents a communication partner in such a relationship.

The property *logicalEndPointId* is a GUID inside the automation system and identifies the *logicalEndPoint*.

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

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

4.3 Physical topology

The physical topology describes the physical view on the communication partners. It represents 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

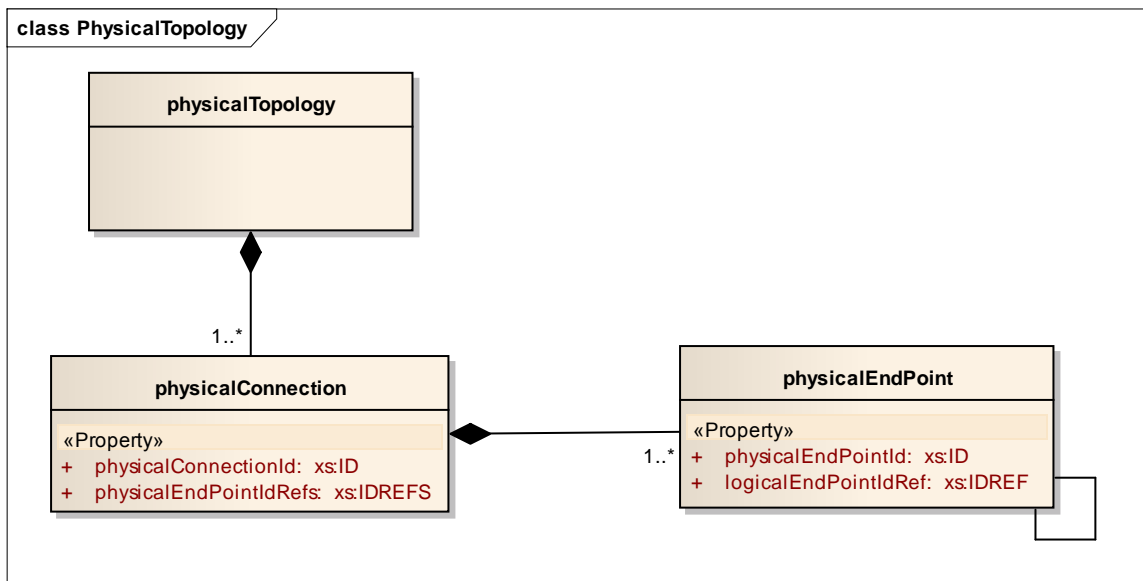


Figure 19 - View on physical topology

4.3.1 Item physicalTopology

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

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

4.3.2 Item physicalConnection

The entity *physicalConnection* describes the communication relation between two physical end points.

The property *physicalConnectionId* is a GUID inside the plant and identifies the *physicalConnection*.

The property *physicalEndPointIdRefs* is an array containing 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.

4.3.3 Item physicalEndPoint

The entity *physicalEndPoint* represents a real communication endpoint in such a relationship.

The property *physicalEndPointId* is a GUID inside the automation system and identifies the *physicalEndPoint*.

The property *logicalEndPointIdRef* is a GUID of a logicalEndPoint related to this *physicalEndPoint*.

Derivations of such end point of a communication may be an Ethernet port.

4.4 Device

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.

4.4.1 Item physicalDevice

Central component is the *physicalDevice*. The attributes allow identifying the instances and sub-instances of it uniquely. Furthermore, there is an aggregation relationship to itself. By means of this

```

classDiagram
    class information {
        «Property»
        + descriptor: xs:string
        + deviceClassification: xs:NMTOKEN
        + installationManual: int
        + networkConfigurationManual: int
        + productDate: xs:date
        + productId: xs:nonNegativeInteger
        + productName: xs:normalizedString
        + productRevision: xs:normalizedString
        + userManual: int
        + vendor: xs:normalizedString
        + vendorId: xs:nonNegativeInteger
    }
    class physicalDevice {
        - subDeviceNumber: xs:nonNegativeInteger
        «Property»
        + IEC81346FunctionReference: xs:normalizedString
        + IEC81346LocationReference: xs:normalizedString
        + IEC81346ProductReference: xs:normalizedString
        + physicalDeviceId: xs:ID
    }
    class logicalDevice {
        - deviceConfig: xs:string
        «Property»
        + deviceDescriptionReference: xs:anyURI
        + logicalDeviceId: xs:ID
        + protocol: xs:normalizedString
        + usedPhysicalEndPointRef: xs:IDREFS
    }
    class physicalEndPointList
    class physicalChannelList
    class deviceResource {
        «Property»
        + resourceId: xs:ID
    }
    class networkData
    class logicalPortList
    class physicalEndPoint {
        «Property»
        + physicalEndPointId: xs:ID
        + logicalEndPointIdRef: xs:IDREF
    }
    class physicalChannel
    class variableList
    class networkDataItem
    class logicalEndPoint {
        «Property»
        + logicalEndPointId: xs:ID
        + logicalEndPointRole: xs:string
    }
    class variable

    information "0..*" --> "1" physicalDevice
    information "0..*" --> "0..1" logicalDevice
    physicalDevice "1" --> "1..*" physicalEndPoint
    physicalDevice "0..*" --> "0..1" physicalChannelList
    physicalDevice "0..*" --> "0..1" deviceResource
    logicalDevice "0..1" --> "1..*" networkData
    logicalDevice "0..1" --> "1..*" logicalPortList
    physicalEndPoint "0..*" --> "1..*" logicalEndPoint
    physicalEndPoint "0..*" --> "1..*" physicalChannel
    deviceResource "0..*" --> "1..*" variableList
    networkData "1..*" --> "1..*" networkDataItem
    logicalPortList "1..*" --> "1..*" logicalEndPoint
    variableList "1..*" --> "1..*" variable

```

The property *physicalDeviceId* is a GUID inside the automation system and identifies the *physicalDevice*. If it is a submodule inside a device, the property *subDeviceNumber* defines the manufacturer specific order of the submodule in the device.

The property *IEC81346LocationReference* describes the location of the *physicalDevice* in the plant according the rules of IEC 81346.

4.4.2 Item Information

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

4.4.3 Item variable

The item *variable* represents a variable e.g. of a PLC program or in another type of controller. The data type is not explicitly modelled here. The *variable* is listed via *variableList*.

The property *variableId* is a GUID inside the automation system and identifies the *variable*.

4.4.4 Item physicalChannel

The item *physicalChannel* represents 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 inside the automation system and identifies the *channel* or *signal*.

4.4.5 Item logicalDevice

The item *logicalDevice* represents 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* aggregates the *networkData*, used for configuration tasks. Also the *logicalEndpoints* are aggregated via the *logicalEndpointList*. In addition to the representation of the logical topology in section 4.2, the *logicalEndPoint* aggregates also the protocol data units, exchanged via the logical connection.

The property *deviceConfig* describes information used during start starting of the *logicalDevice*. They may be exchanged at runtime between different communication partners, depending on the communication system.

The property *deviceDescriptionReference* is a reference to an external device description file.

The property *logicalDeviceId* is a GUID inside the automation system and identifies the *logicalDevice*.

The property *protocol* describes the used communication protocol, e.g. PROFINET or PROFIBUS etc.

The property *usedPhysicalEndPointIdRef* represents a list of GUID with associated *physicalEndpoints*. This means, that such end point will be used by the *logicalDevice*.

4.4.6 Item pduList

The item *pduList* represents the allowed Protocol Data Units exchanged via the *logicalEndPoint*.

4.4.7 Item pdu

The item *pdu* represents one specific Protocol Data Unit to be exchanged by means of a *logicalConnection*. It aggregates the *protocolData* and the payload. The payload can be specialised into *processDataItem* or *parameterItem*.

4.4.8 Item protocolData

The item *protocolData* can 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.

4.4.9 Item payload

The item *payload* represents the *processDataItem* and *parameterItem* as a generalized abstract item. The optional property *address* may be used for address or identification reasons of a *dataItem*.

4.4.10 Item processDataItemList

This item is a pattern to enumerate all allowed *processDataItems*.

4.4.11 Item parameterItemList

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

4.4.12 Item *dataItem*

The item *dataItem* represents a data object communicated via the *logicalConnection*.

The property *octetOffset* defines the octet offset of the data object in the PDU *payload*. The counting starts with the first octet of the *payload* starting with index '0' for first octet to send / receive.

The property *bitOffset* defines 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 have to be added in order to calculate the bit position.

The property *numberOfBits* defines the length of data object in bits.

4.4.13 Item *processDataItem*

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

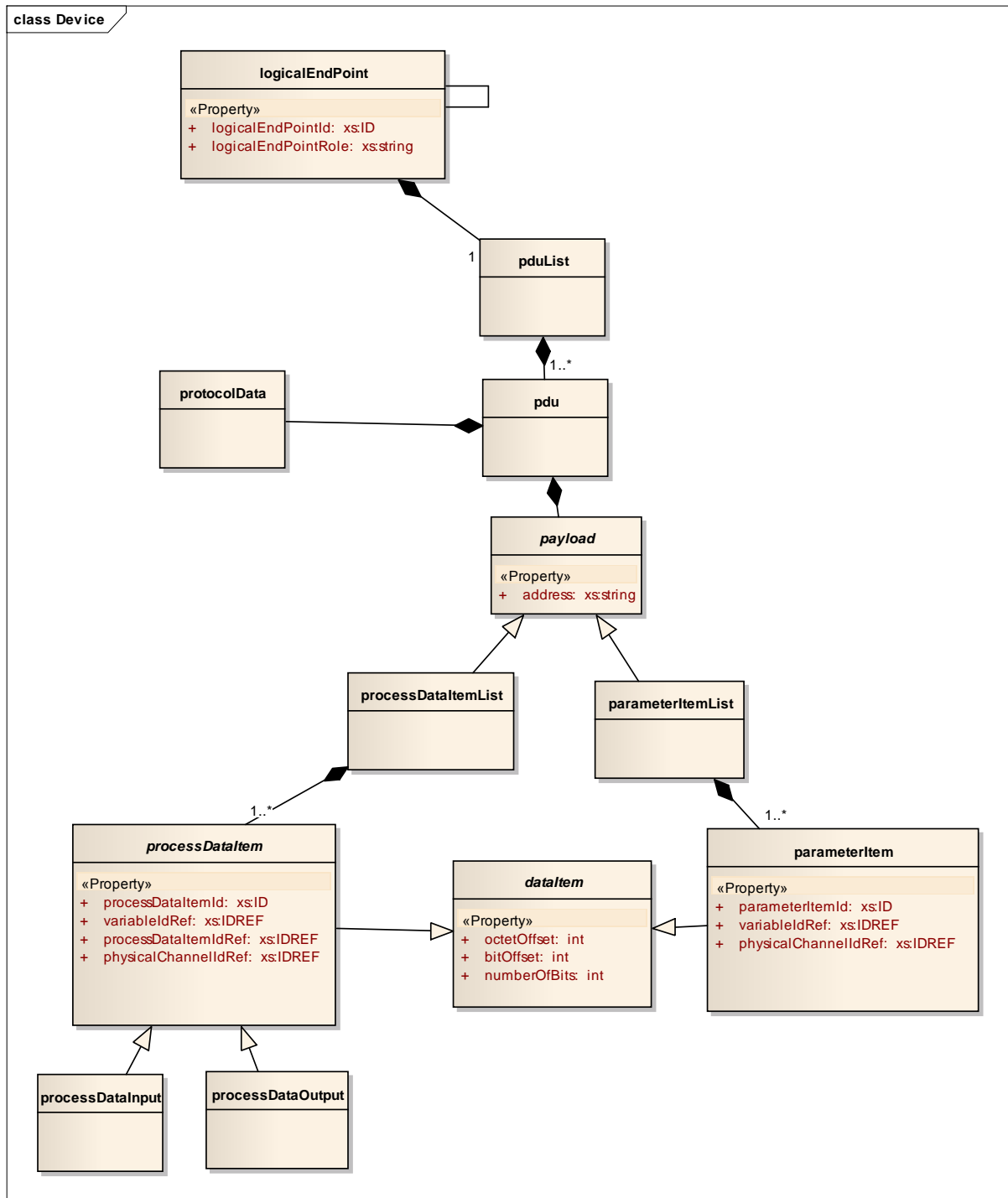


Figure 21 - Part 2 of the device model

The property *processDataItemId* is a GUID inside the automation system and identifies the *processDataItem*.

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

The property *processDataItemIdRef* is 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 PLC or not a controller.

The property *physicalChannelIdRef* is the reference of the *processDataItem* to a *physicalChannel* of a device.

4.4.14 Item *processDataInput*

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

4.4.15 Item *processDataOutput*

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

4.4.16 Item *parameterItem*

The item *parameterItem* represents 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 identifies the *parameterItem*.

The property *variableItemIdRef* is the reference of a variable of the PLC program. The reference is set, if the *physicalDevice* is a PLC or controller.

The property *physicalChannelIdRef* is the reference of the *processDataItem* to a *physicalChannel* of a device.

5 Representation within AutomationML

5.1 Overview 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 Section 3 and Section 4.

5.1.1 General mapping rules

For representation of the different entities named in in Section 3 and Section 4 the following mapping rules will be applied. Thereby it is distinguished which how the semantic of an object is represented and an related entity is modelled.

Table 2: Mapping rules

Information entity of UML model	Semantik 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	Interface
physicalChannel	InterfaceClass SignalInterface	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
processDataItem	Implicitly modelled by hosting InternalElement	
parameterItem	Implicitly modelled by hosting InternalElement	
DataItem	InterfaceClass DatagrammObject	Interface
processDataItem	InterfaceClass DatagrammObject	Interface
parameterItem	InterfaceClass DatagrammObject	Interface

5.1.2 Basics

Basic 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 of relevant role classes and interface classes for the special application cases from them following the definition of role class and interface class application defined in Part 1 of the AutomationML specification [4].

The AutomationML communication role class library will contain roles dedicated to identify internal elements as physical devices, physical connections and physical networks as well as internal elements 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 the AutomationML role class library and the AutomationML interface class library are depicted in Figure 22.

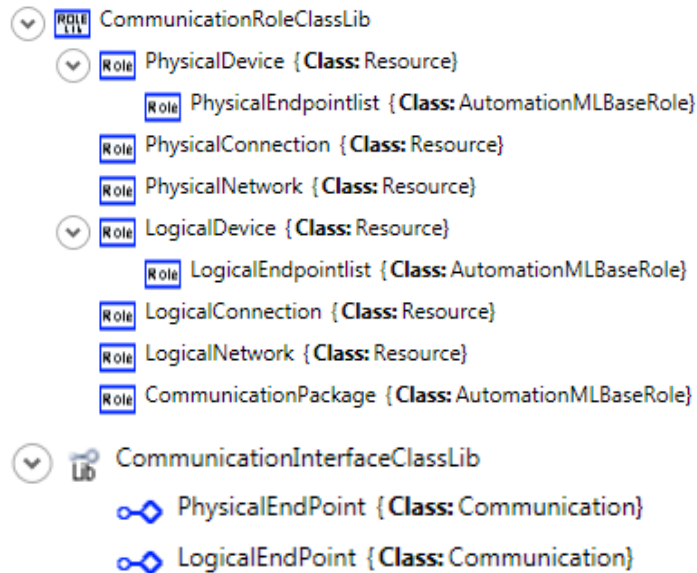


Figure 22 - Communication role class library and communication interface class library

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

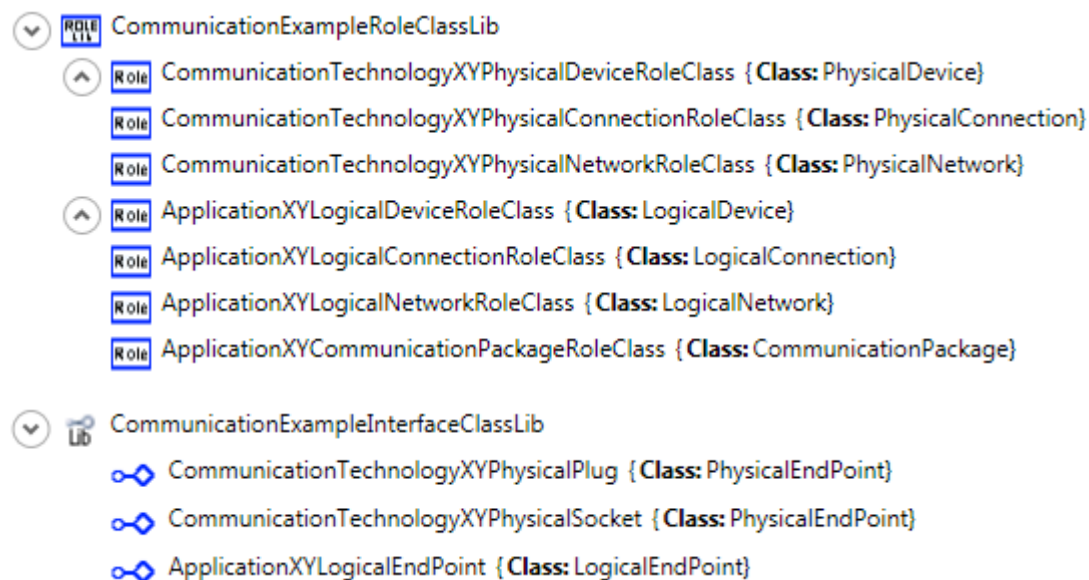


Figure 23 - Derived role class libraries and interface class libraries for a special example

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

Each physical connection object will be equipped by as much physical end point objects as the connection can connect at physical devices. Usually in case of physical wiring these are two end point objects. Each logical connection object will be equipped by as much logical end point objects as the connection can connect at 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 be modelled now. Therefore, all necessary physical and logical devices are instantiated as <InternalElement>s in an appropriate <InstanceHierarchy>. Especially the hierarchical structure of the modelled system has to be reflected / preserved. This is especially valide 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 have to be completed and filled with values.

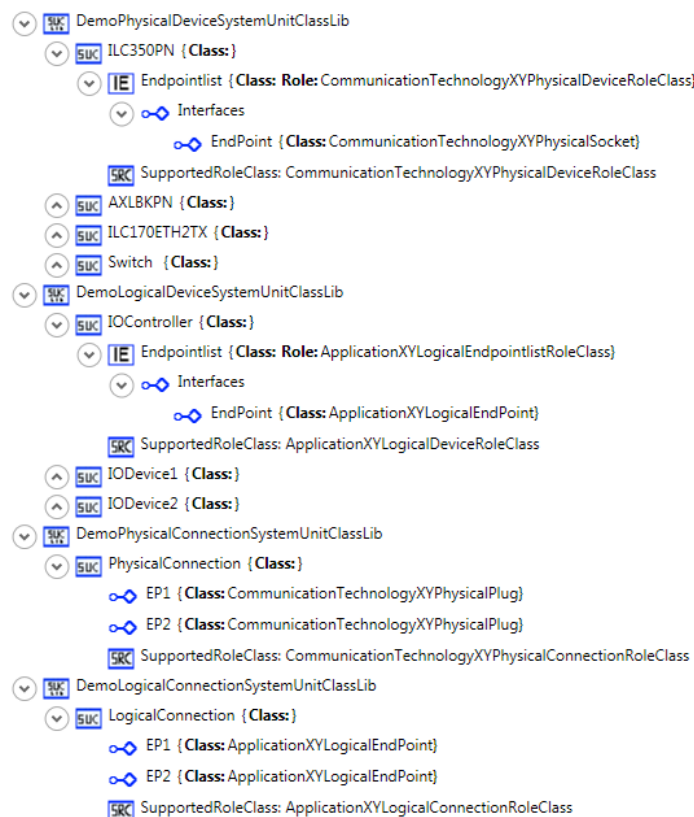


Figure 24 - SystemUnitClassLib examples for communication system modelling

If 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.

If 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 map also the logical end points to physical end points implementing the related connections internal link objects are used connecting physical end points with logical end points.

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

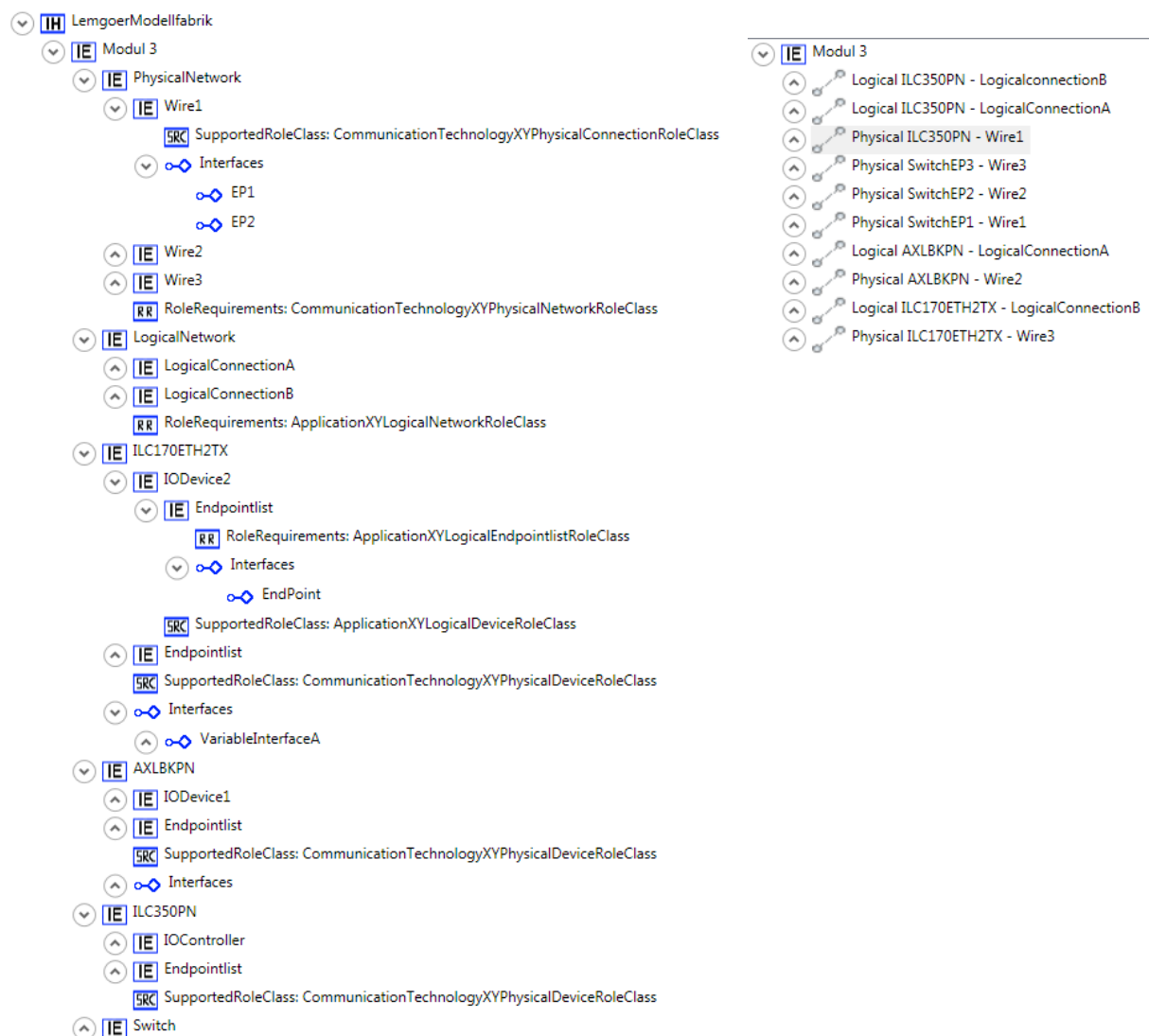


Figure 25 - Final network model example

It has to be noted that depending on the application case parts of the modelling process named above can be omitted. In case of modelling only physical networks, the definition of logical network entity role classes, interface classes, and system unit classes can be neglected, to name one example. In addition a sequential system modelling starting with either the physical or the logical network and adding the other one later on is possible.

5.2 Basic communication role class library

5.2.1 General

This clause defines a base library of essential standard role classes required for the modelling of core communication concepts (Figure 26 - Figure 28). All roles are technology independent but the basis for technology dependent libraries. Details of each role class are given in 5.2.2 to 5.2.9

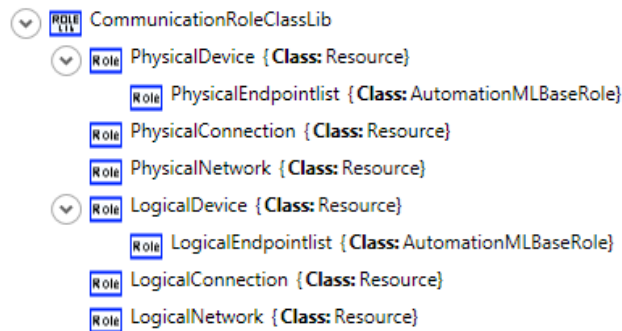


Figure 26 - Basic communication role class library

RoleClassLib (3)			
	= Name	{ De...	{ Version
1	CommunicationRoleClassLib		1.0
	RoleClass (6)		
	= Name	= RefBaseClassPa...	{ RoleClass
1	PhysicalDevice	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	RoleClass Name...
2	PhysicalConnection	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
3	PhysicalNetwork	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
4	LogicalDevice	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	RoleClass Name...
5	LogicalConnection	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
6	LogicalNetwork	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	

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>
  
```

Figure 28 - XML text of the communication role class library

5.2.2 RoleClass 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, PLCs or Profibus coupler.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Resource	
Attributes	None	

5.2.3 RoleClass PhysicalEndpointlist*Table 4 – RoleClass PhysicalEndpointlist*

Class name	PhysicalEndpointlist	
Description	<p>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 children of a physical device and all physical communication interfaces shall be children of the port list object.</p> <p>This allows modelling of physical devices with multiple communication interfaces, e.g. a profibus and a profinet connector.</p>	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

5.2.4 RoleClass 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	

5.2.5 RoleClass 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	

5.2.6 RoleClass 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	

5.2.7 RoleClass LogicalEndpointlist*Table 8 - RoleClass LogicalEndpointlist*

Class name	LogicalEndpointlist	
Description	<p>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.</p> <p>This allows modelling of logical devices with multiple communication interfaces.</p>	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Attributes	None	

5.2.8 RoleClass 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	

5.2.9 RoleClass 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	

5.3 Basic communication interface class library

5.3.1 General

This clause defines a base library of essential standard role classes required for the modelling of core communication concepts (Figure 29 - Figure 31). All roles are technology independent but the basis for technology dependent libraries. Details of each interface class are given in 5.3.2 and 5.3.3.

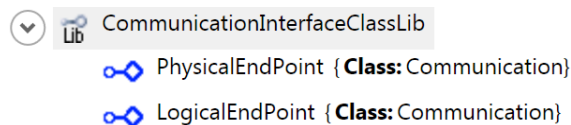


Figure 29 - Basic communication interface class library

InterfaceClassLib	
Name	CommunicationInterfaceClassLib
Version	1.0
InterfaceClass	
Name	PhysicalEndPoint
RefBaseClassPath	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
InterfaceClass	
Name	LogicalEndPoint
RefBaseClassPath	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication

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>
  
```

Figure 31 - XML text of the communication interface class library

5.3.2 InterfaceClass 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	

5.3.3 InterfaceClass 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	

5.4 Steps to model technology specific libraries

5.4.1 General

The described basic role and interface classes are still independent from any technology. In order to model communication systems based on concrete technologies as *Profinet*, *EthernetIP*, *Interbus* or *Sercos*, corresponding technology specific classes have to be derived. This section is about the definition of technology specific libraries in 3 steps. This stepwise approach is intended for a stepwise development of the communication system model. Logical and physical networks must 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 whitepaper.

5.4.2 Step 1: Development of technology specific role classes

In a first step, a technology specific role class library has to be created whereas 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).

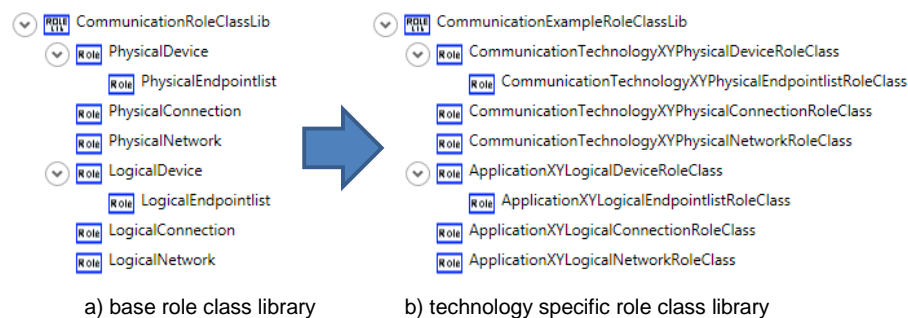


Figure 32 - Derivation of a technology specific role class library out of the base role class library

In practice, for each concrete communication technology as *Profinet*, *EthernetIP*, *Interbus* or *Sercos*, an own library has to be developed based on the physical network related roles. In addition for each concrete application as device configuration service, drive control, line station control, etc. an own library can come up based on the logical network related roles.

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

5.4.3 Step 2: Development of technology specific interface classes

In a second step, a technology specific interface class library has to be created whereas each of the specific communication interface classes is derived from the corresponding basic class. Since most communications base on plugs and sockets, in this example, the class *PhysicalEndPoint* leads to two classes. In this example, the resulting library is named *CommunicationExampleInterfaceClassLib*.

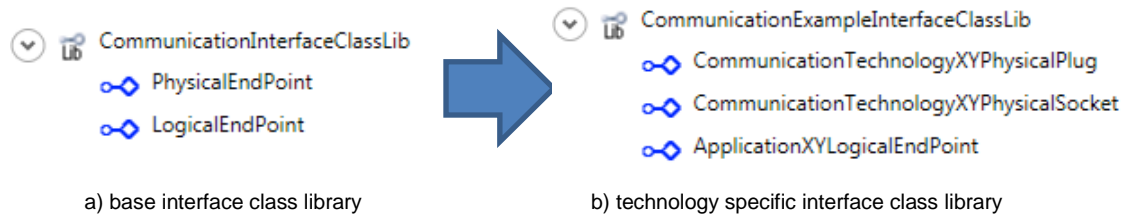


Figure 33 - Derivation of a technology specific role class library out of the base role class library

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

5.4.4 Step 3: Development of system unit class libraries

Based on the technology specific role and interface class libraries, all needed physical devices and physical connections have to be modelled as *<SystemUnitClass>* in a *<SystemUnitClassLib>* which reference 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. Furthermore, the connection classes are modelled. The resulting libraries are depicted in Figure 34.

The naming of system unit classes and its 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.

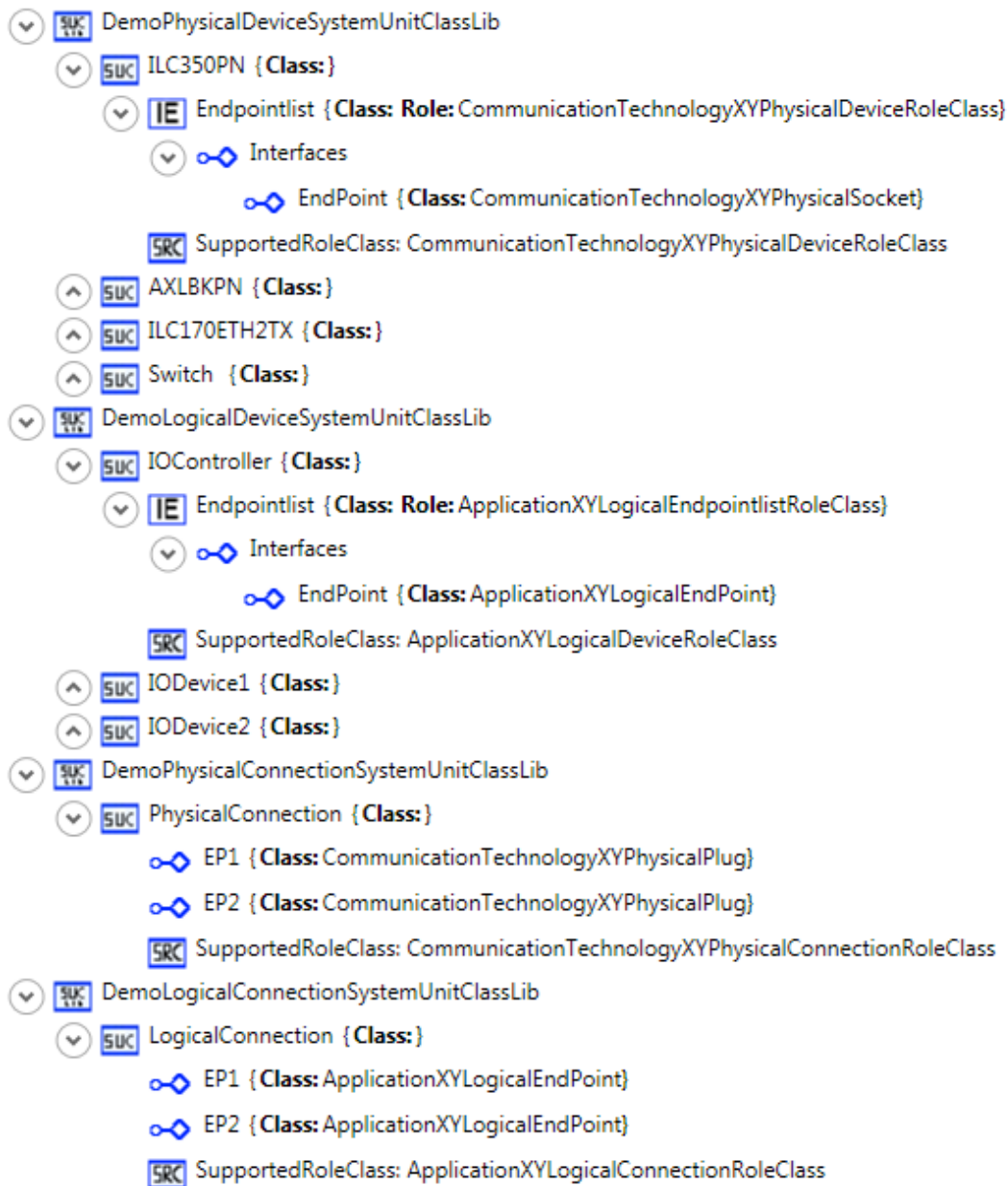


Figure 34 - Technology specific <SystemUnitClassLib>s

5.4.5 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 have to be instantiated as new <InternalElement>s in the <InstanceHierarchy> in their needed hierarchical order. All relevant attributes get relevant values in order to describe individual properties of the devices.

5.4.6 Step 5: Modelling the connections

The last step 5 serves for the interconnection of the devices. The physical and logical network has to be modelled each as own <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 have to 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.

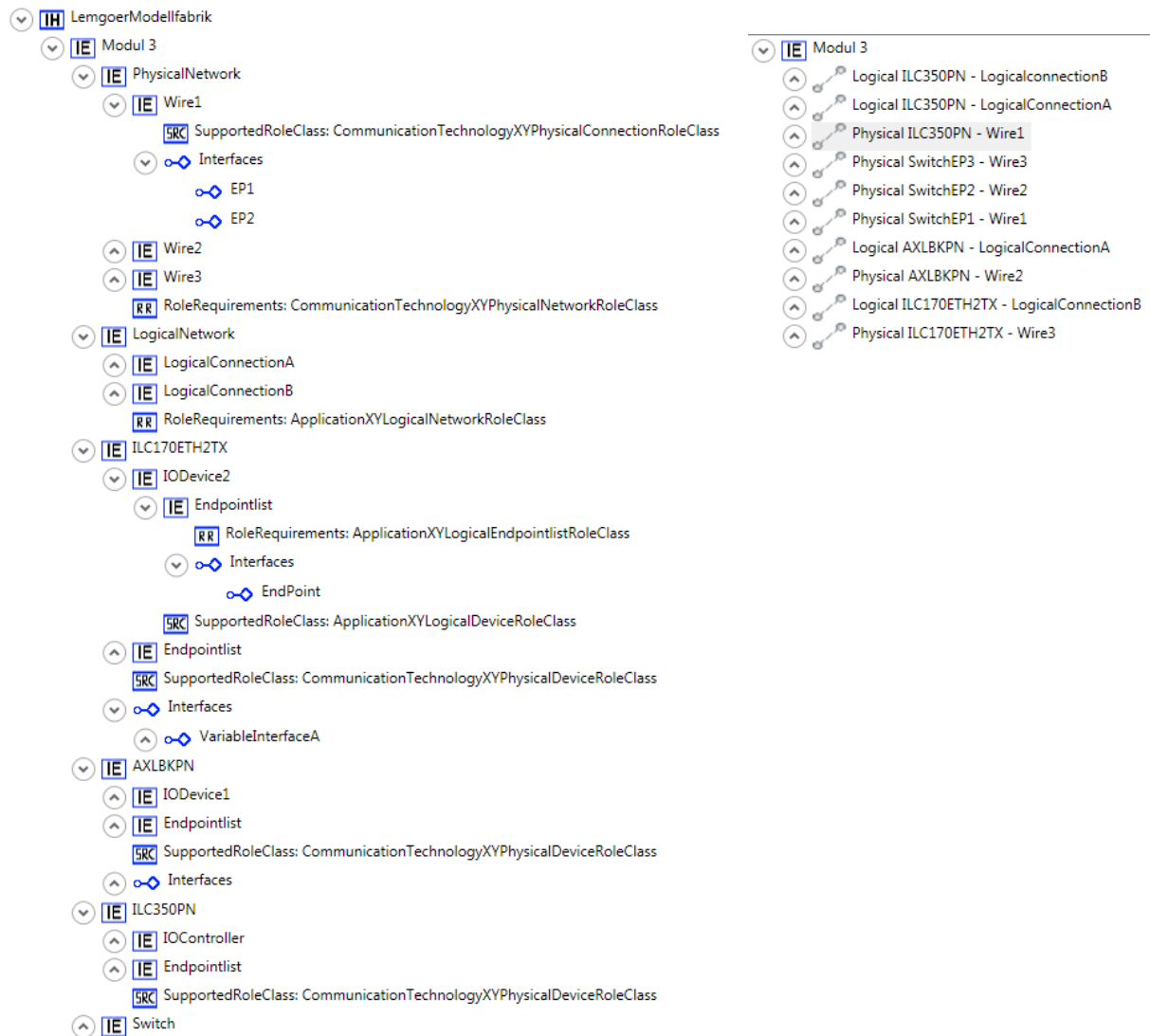


Figure 35 - Technology specific communication network

5.5 PDU modelling

Following Section 3.2.4 and 4.4 logical devices will exchange communication data packages (PDU) via its logical interfaces.

This clause defines an extension of the communication libraries of essential standard role classes and essential interfaces classes as well as represents the necessary modelling steps for modelling PDUs within a communication system model required to model communication data packages.

5.5.1 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 38.

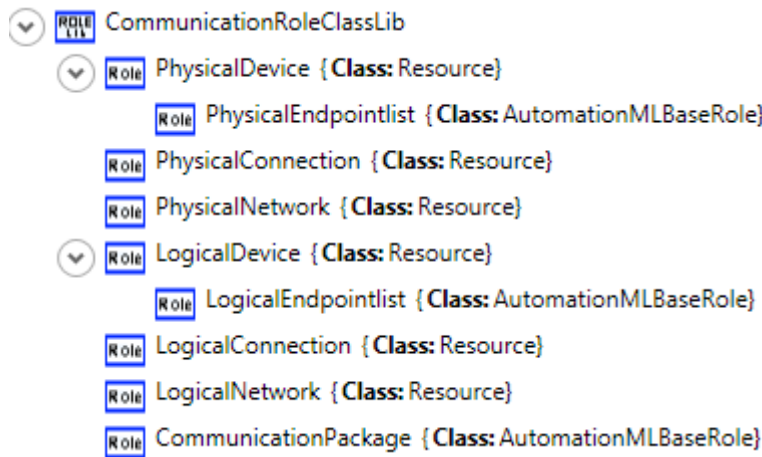


Figure 36 - Extended communication role class library

= Name		D Version		RoleClass	
1	CommunicationRoleClassLib		1.0	RoleClass (7)	
		= Name		= RefBaseClassPath	RoleClass
		1 PhysicalDevice		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	RoleClass
					= Name PhysicalEndpointlist
					= RefBaseClassPa... AutomationMLBaseRoleClassLib/Automatio nMLBaseRole
		2 PhysicalConnection		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	
		3 PhysicalNetwork		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	
		4 LogicalDevice		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	RoleClass
					= Name LogicalEndpointlist
					= RefBaseClassPa... AutomationMLBaseRoleClassLib/Automatio nMLBaseRole
		5 LogicalConnection		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	
		6 LogicalNetwork		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole/Resource	
		7 CommunicationPackage		AutomationMLBaseRoleClassLib/ AutomationMLBaseRole	

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>
  
```

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

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	

5.5.2 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 41.

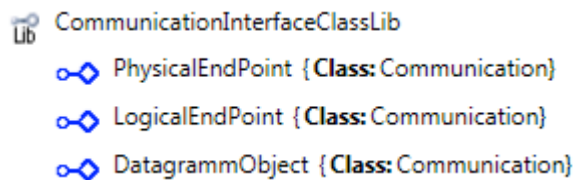


Figure 39 - Extended communication interface class library

CommunicationInterfaceClassLib	1.0	InterfaceClass (3)
= Name		= RefBaseClassPa...
1	PhysicalEndPoint	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
2	LogicalEndPoint	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication
3	DatagrammObject	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication

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>
  
```

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

Table 14 - InterfaceClass DatagrammObject

Class name	DatagrammObject	
Description	<p>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.</p> <p>It represents a consistent information set exchanged which corresponds to the information set represented by a PLCOpenXMLInterface as defined in Part 4 of the AutomationML specification.</p>	
Parent Class	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/Communication	
Attributes	None	

5.5.3 Steps to model technology specific libraries

5.5.3.1 General

As described in Section 5.4 the extended role and interface classes are still independent from any technology. In order to model communication systems based on concrete technologies as *Profinet*, *EthernetIP*, *Interbus* or *Sercos*, corresponding technology specific classes have to be derived. This section is about the extension of the definition of technology specific libraries given in Section 5.4 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 whitepaper.

5.5.3.2 Step 1: Development of technology specific role classes

The first step given in Section 5.4.2 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.

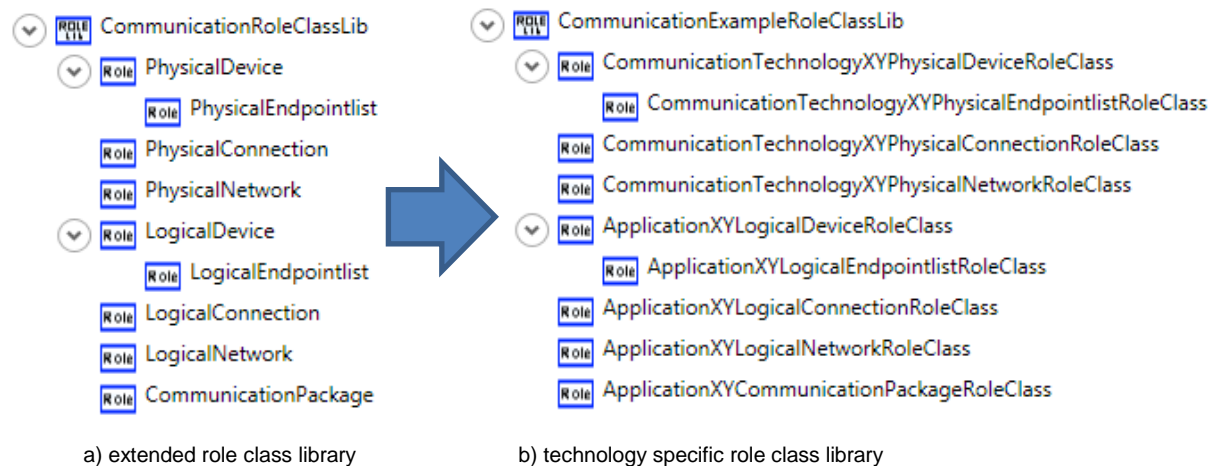


Figure 42 - Derivation of a technology specific role class library out of the extended role class library

5.5.3.3 Step 2: Development of technology specific interface classes

The second step given in Section 5.4.3 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 49. Thereby an *ApplicationXYDatagrammObject* interface class is derived from *DatagrammObject* interface class.

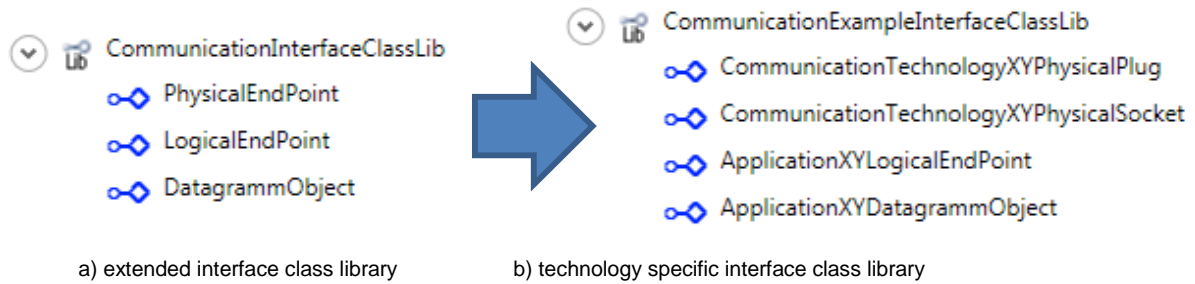


Figure 43 - Derivation of a technology specific interface class library out of the extended interface class library

5.5.3.4 Step 3: Development of system unit class libraries

The third step given in Section 5.4.4 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.

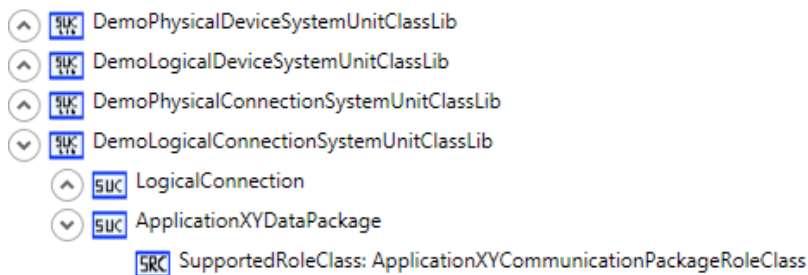


Figure 44 - Technology specific extended <SystemUnitClassLib>s

5.5.3.5 Step 4 and 5 – Modelling of communication network

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

5.5.3.6 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.

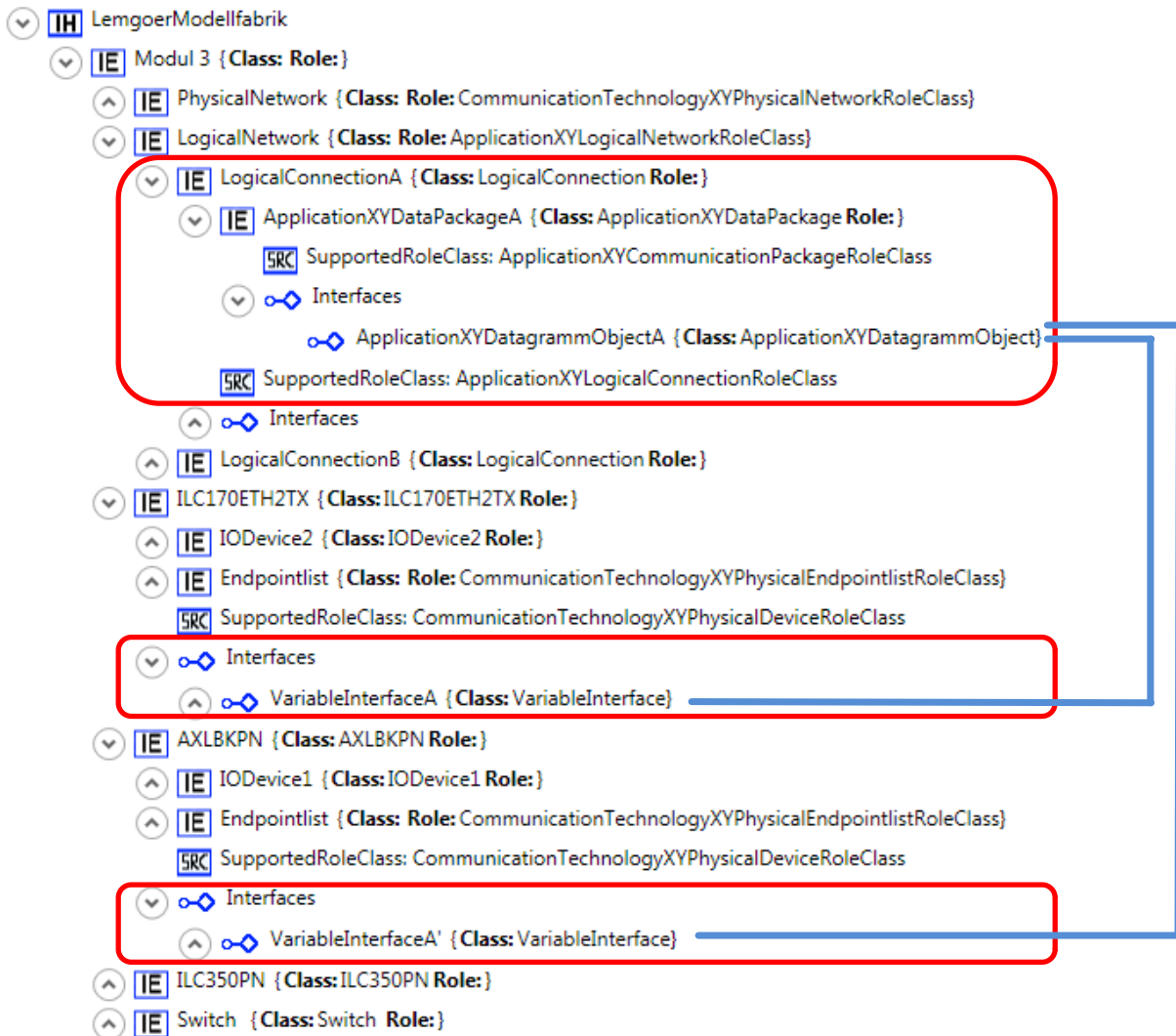


Figure 45 - Technology specific communication network with communication package models

5.6 References to attributes

As given in sections 5.4.4. and 5.5.3.6 communication device, interface, connection, PDU, and datagram object related attributes are modelled by means of CAEX attributes. Within the following table 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 <i>address</i> 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 <i>address offset</i> specifies which bit in the I/O memory map	UINT	Not for logical and physical devices	8.5

Attribute type name	Description	Data Type, value range	Use limitations	Example
	of the assigned I/O unit the I/O signal is mapped to. The offset of the first I/O is 0.			
BitCount	The attribute <i>bit count</i> 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 <i>IEC data type</i> , if available.	UINT >0		8
BitOffset	This attribute specifies which bit of a certain byte in the I/O memory map of the assigned I/O unit the I/O signal is mapped to. The offset of the first bit in the first byte I/O is 0.	Derived from AddressOffset	Not for logical and physical devices	5
BitOrder	The attribute <i>bit order</i> 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 I/O memory map of the assigned I/O unit the I/O signal is mapped to. The octet offset of the first I/O is 0. The octets are counted from the lowest octet.	Derived from AddressOffset	Not for logical and physical devices	8
Direction	The attribute <i>direction</i> specifies the direction of the signal, either from the controller perspective (hard-wired I/Os) 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 <i>IEC data type</i> 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,		WORD

Attribute type name	Description	Data Type, value range	Use limitations	Example
		ULINT, REAL, LREAL, TIME, LTIME, DATE, LDATE		
InitialValue	The parameter <i>initial value</i> specifies the I/O signal value to be used at start.	Bytearray according to the IECDataType		0
LogicalUnit	The attribute <i>logical unit</i> 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 <i>maximum physical value</i> . The logical values offer a way to access the I/O signals by using logical quantities rather than physical. The attribute is only applicable for analog signals.	String ²	Only for logical end points	30
MaxPhysicalValue	This attribute specifies the physical value that will correspond to the <i>maximum bit value</i> . The physical value directly corresponds to the value of the I/O signal that this system attribute corresponds to, i.e. the amount of voltage or current given by a sensor. The attribute <i>maximum physical value</i> is only applicable for analog signals.	String ¹	Only for logical and physical end points and physical connections	20
MinLogicalValue	Minimum logical value of a measured physical value	String ²	Only for logical end points	10
MinPhysicalValue	Minimum physical	String ²	Only for logical	4

¹ The format of the string content shall conform to the data type given in the attribute IECDataType.

² The format of the string content shall conform to the data type given in the attribute IECDataType.

Attribute type name	Description	Data Type, value range	Use limitations	Example
alue	value of a measured physical value		and physical end points and physical connections	
PhysicalUnit	<p>The attribute <i>physical unit</i> specifies the unit of the I/O signal. e.g. "mA", "V"</p> <p>The attribute <i>physical unit</i> is only applicable for analog signals.</p>	String	Only for logical and physical end points and physical connections	mA

5.7 Usage of Metadata

Metadata are modelled according to IEC 62424:2014 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 (Table 16).

Table 16 - Field SourceDocumentatInformation according to communication related libraries

SourceDocumentInformation	
= OriginID	DKE
= OriginName	DKE
= OriginVersion	3.0
= LastWritingDateTime	2013-11-14T12:00:00.0Z

Version and revision related information shall be modelled according to IEC 62424:2014.

6 Examples

Within the following two communication system examples are modelled to give reference points for the use of the communication system modelling described above.

6.1 Lab size manufacturing system model

Within the following the communication system part of a lab size manufacturing system model depicted in Figure 46 is modelled.

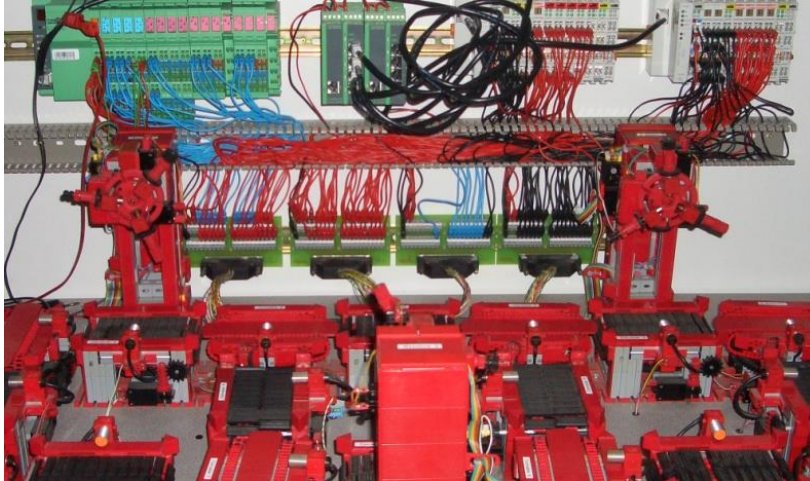


Figure 46 - Lab size manufacturing system model

This communication system consists of three logically connected control devices hosting the control application, a switch and three Ethernet wires establishing the physical network. The physical and logical topologies of the network example are depicted in Figure 47.

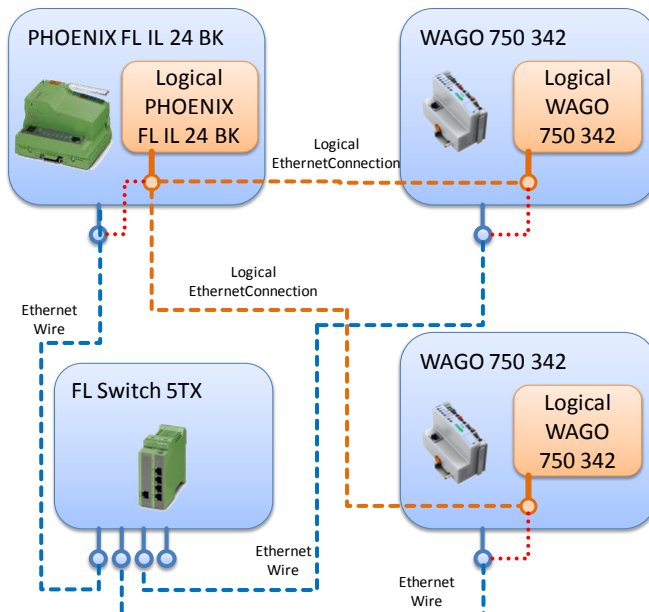


Figure 47 - Physical and logical topology of network example

The modelling steps presented in Section 5.4 are used to model the following artefacts of the communication system. The complete *.aml file is given on the web page of AutomationML under <https://www.automationml.org/o.red.c/dateien.html>.

6.1.1 Role classes

The following role class library has been derived following section 5.4.2 from the basic roles classes defined in section 5.2.

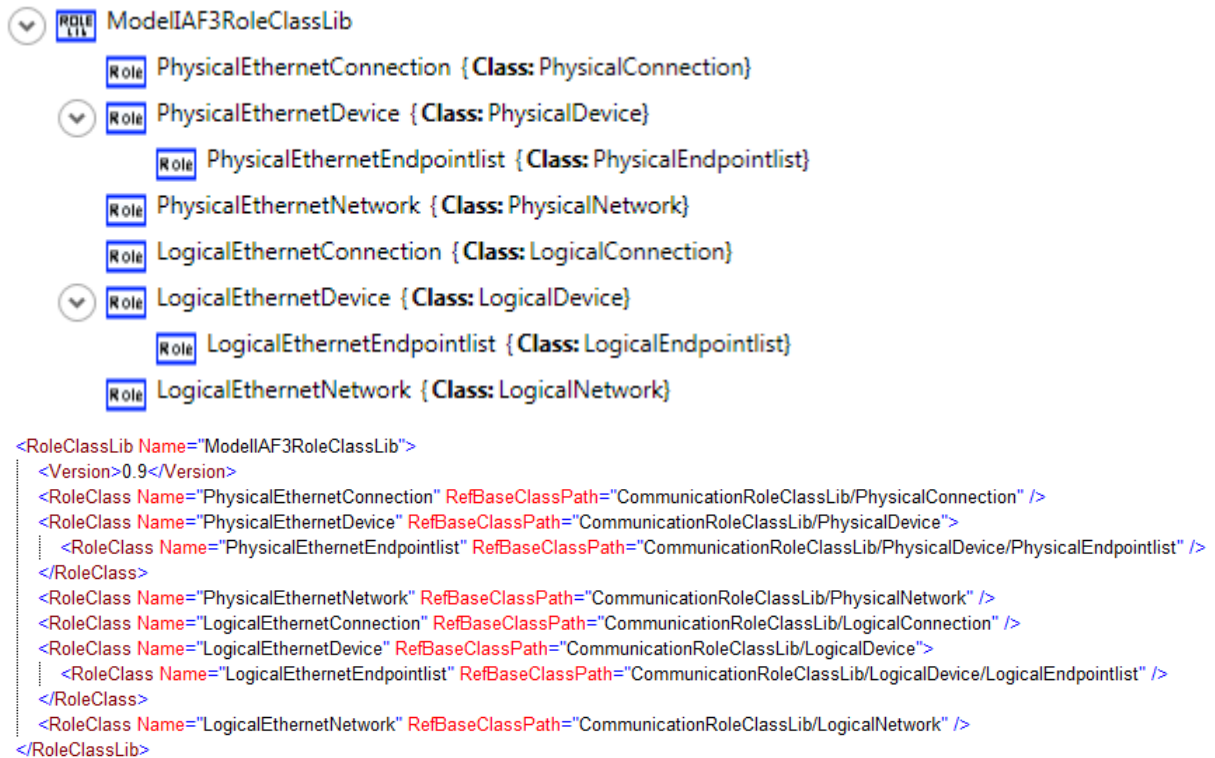


Figure 48 - Role class library for plant model example

Table 17 - Role PhysicalEthernetConnection

Name	PhysicalEthernetConnection
Description	Physical Network Connections specifically for Ethernet Networks
ParentClass	CommunicationRoleClassLib/PhysicalConnection

Table 18 - Role PhysicalEthernetDevice

Name	PhysicalEthernetDevice
Description	Physical Network Devices specifically for Ethernet Network
ParentClass	CommunicationRoleClassLib/PhysicalDevice

Table 19 - Role PhysicalEthernetEndpointlist

Name	PhysicalEthernetEndpointlist
Description	Physical Network Devices Port List specifically for Ethernet Networks
ParentClass	CommunicationRoleClassLib/PhysicalEndpointlist

Table 20 - Role PhysicalEthernetNetwork

Name	PhysicalEthernetNetwork
Description	Physical Ethernet Network

ParentClass	CommunicationRoleClassLib/PhysicalNetwork
--------------------	---

Table 21 - Role LogicalEthernetConnection

Name	LogicalEthernetConnection
Description	Logical Network Connection specifically for Ethernet Network
ParentClass	CommunicationRoleClassLib/LogicalConnection

Table 22 - Role LogicalEthernetDevice

Name	LogicalEthernetDevice
Description	Logical Network Device specifically for Ethernet Network
ParentClass	CommunicationRoleClassLib/LogicalDevice

Table 23 - Role LogicalEthernetEndpointlist





Name	LogicalEthernetEndpointlist
Description	Logical Network Device Port List specifically for Ethernet Network
ParentClass	CommunicationRoleClassLib/LogicalEndpointlist

Table 24 - Role LogicalEthernetNetwork

Name	LogicalEthernetNetwork
Description	Logical Ethernet Network specifically for Ethernet Networks
ParentClass	CommunicationRoleClassLib/LogicalNetwork

6.1.2 Interface classes

The following interface class library has been derived following section 5.4.3 from the basic interface classes defined in section 5.2.

 **ModelIAF3CommunicationInterfaceClassLib**
 EthernetPlug { **Class:** PhysicalEndPoint}
 EthernetSocket { **Class:** PhysicalEndPoint}
 LogicalEthernetEndPoint { **Class:** LogicalEndPoint}

```

<InterfaceClassLib Name="ModelIAF3CommunicationInterfaceClassLib">
  <Version>0.9</Version>
  <InterfaceClass Name="EthernetPlug" RefBaseClassPath="CommunicationInterfaceClassLib/PhysicalEndPoint" />
  <InterfaceClass Name="EthernetSocket" RefBaseClassPath="CommunicationInterfaceClassLib/PhysicalEndPoint" />
  <InterfaceClass Name="LogicalEthernetEndPoint" RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
</InterfaceClassLib>

```

Figure 49 - Interface class library for plant model example

Table 25 - InterfaceClass EthernetPlug

Name	EthernetPlug
Description	Network Plug specifically for Ethernet Networks

ParentClass	CommunicationInterfaceClassLib/PhysicalEndPoint
--------------------	---

Table 26 - InterfaceClass EthernetSocket

Name	EthernetSocket
Description	Bush specifically for Ethernet Networks
ParentClass	CommunicationInterfaceClassLib/PhysicalEndPoint

Table 27 - InterfaceClass LogicalEthernetEndPoint

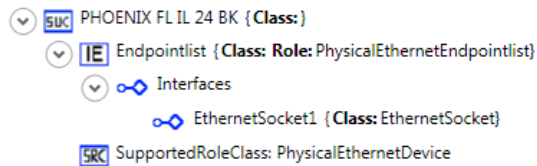
Name	LogicalEthernetEndPoint
Description	Terminal Points specifically for Logical Ethernet Network, relates to EthernetPlug and Socket in physical Ethernet Network
ParentClass	AutomationMLInterfaceClassLib/AutomationMLBaseInterface

6.1.3 System Unit Classes

The following system unit class libraries have been derived following section 5.4.4.

<ul style="list-style-type: none"> ModelIAF3 <ul style="list-style-type: none"> PHOENIX FL IL 24 BK {Class:} WAGO 750 342 {Class:} FL Switch 5TX {Class:} Ethernet Wire {Class:} LogicalModelIAF3 <ul style="list-style-type: none"> Logical PHOENIX FL IL 24 BK {Class:} Logical WAGO 750 342 {Class:} Logical EthernetConnection {Class:} 	<pre> <SystemUnitClassLib Name="ModelIAF3"> <Version>0.9</Version> <SystemUnitClass Name="PHOENIX FL IL 24 BK"> <SystemUnitClass Name="WAGO 750 342"> <SystemUnitClass Name="FL Switch 5TX"> <SystemUnitClass Name="Ethernet Wire"> </SystemUnitClassLib> <SystemUnitClassLib Name="LogicalModelIAF3"> <Version>0.9</Version> <SystemUnitClass Name="Logical PHOENIX FL IL 24 BK"> <SystemUnitClass Name="Logical WAGO 750 342"> <SystemUnitClass Name="Logical EthernetConnection"> </SystemUnitClassLib> </pre>
--	---

Figure 50 - System unit class library for plant model example



rate of transmission	
Name	rate of transmission
Description	
Value	100
Default Value	10
Unit	MBit/s
DataType	xs:integer
supporting rail assembly possible	
quantity the RJ45-Ports	
manufacturer's item number	
manufacturer's name	

```

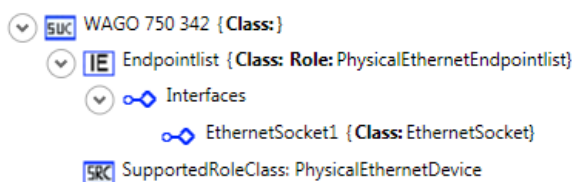
<SystemUnitClass Name="PHOENIX FL IL 24 BK">
  <Attribute Name="rate of transmission" AttributeDataType="xs:integer" Unit="MBit/s">
    <DefaultValue>10</DefaultValue>
    <Value>100</Value>
  </Attribute>
  <Attribute Name="supporting rail assembly possible" AttributeDataType="xs:boolean">
    <Value>true</Value>
  </Attribute>
  <Attribute Name="quantity the RJ45-Ports" AttributeDataType="xs:integer">
    <Value>1</Value>
  </Attribute>
  <Attribute Name="manufacturer's item number" AttributeDataType="xs:string">
    <Value>2831057</Value>
  </Attribute>
  <Attribute Name="manufacturer's name" AttributeDataType="xs:string">
    <Value>Phoenix Contact</Value>
  </Attribute>
  <InternalElement Name="Endpointlist" ID="{ebdc27ea-dfea-4ed0-92bf-827f8c8d3075}">
    <ExternalInterface Name="EthernetSocket1" ID="{32170930-8087-4c7d-91c9-a0192a90951b}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <RoleRequirements RefBaseRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice/PhysicalEthernetEndpointlist" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice" />
</SystemUnitClass>
  
```

Figure 51 - System unit class for physical device example 1

Table 28 - SystemUnitClass PHOENIX FL IL 24 BK

Name	PHOENIX FL IL 24 BK	
Description	Bus Coupler Modbus/TCP	
Attributes	Name	transfer rate
	Value	100
	Default Value	10
	Unit	Mbit/s
	DataType	xs:integer
	Name	possibility of assembling supporting rails
	Value	1
	DataType	xs:boolean
	Name	quantity of RJ45-Ports
	Value	1
	DataType	xs:integer
	Name	manufacturer's item number

	Value	2831057
	DataType	xs:string
	Name	manufacturer
	Value	Phoenix Contact
Endpointlist	Role	ModellAF3RoleClassLib/PhysicalEthernetEndpointlist
	EthernetSocket1ParentClass	ModellAF3CommunicationInterfaceClassLib/EthernetSocket
SupportedRoleClass	ModellAF3RoleClassLib/PhysicalEthernetDevice	



transmission medium	
Name	transmission medium
Description	
Value	Twisted Pair S-UTP 100 Ohm Cat 5
Default Value	
Unit	
DataType	xs:string
rate of transmission	
supporting rail assembly possible	
quantity the RJ45-Ports	
manufacturer's name	

```

<SystemUnitClass Name="WAGO 750 342">
  <Attribute Name="transmission medium" AttributeDataType="xs:string">
    <Value>Twisted Pair S-UTP 100 Ohm Cat 5</Value>
  </Attribute>
  <Attribute Name="rate of transmission" AttributeDataType="xs:integer" Unit="MBit/s">
    <Value>10</Value>
  </Attribute>
  <Attribute Name="supporting rail assembly possible" AttributeDataType="xs:boolean">
    <Value>true</Value>
  </Attribute>
  <Attribute Name="quantity the RJ45-Ports" AttributeDataType="xs:integer">
    <Value>1</Value>
  </Attribute>
  <Attribute Name="manufacturer's name" AttributeDataType="xs:string">
    <Value>Wago</Value>
  </Attribute>
  <InternalElement Name="Endpointlist" ID="{7a12346e-7f20-4e5c-8db2-a9ccc6e44145}">
    <ExternalInterface Name="EthernetSocket1" ID="{868d597f-e838-42d8-af70-8d0df54aa15c}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <RoleRequirements RefBaseRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice/PhysicalEthernetEndpointlist" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice" />
</SystemUnitClass>

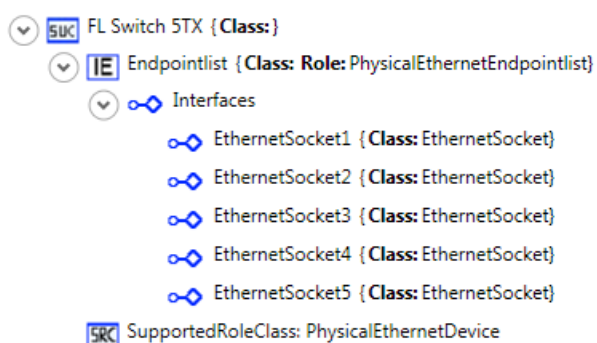
```

Figure 52 - System unit class for physical device example 2

Table 29 - SystemUnitClass WAGO 750 342

Name	WAGO 750 342	
Description	Bus Coupler Modbus/TCP	
Attributes	Name	transmission medium
	Value	Twisted Pair S-UTP 100 Ohm Cat 5

	Data Type	ss:string
	Name	transfer rate
	Value	100
	Default Value	10
	Unit	Mbit/s
	DataType	xs:integer
	Name	possibility of assembling supporting rails
	Value	1
	DataType	xs:boolean
	Name	quantity of RJ45-Ports
	Value	1
	DataType	xs:integer
	Name	manufacturer
	Value	Wago
	DataType	xs:string
Endpointlist	Role	ModelIIAF3RoleClassLib/PhysicalEthernetEndpointlist
	EthernetSocket1 ParentClass	ModelIIAF3CommunicationInterfaceClassLib/ EthernetSocket
SupportedRole Class	ModelIIAF3RoleClassLib/PhysicalEthernetDevice	



rate of transmission	
Name	rate of transmission
Description	
Value	100
Default Value	10
Unit	MBit/s
DataType	xs:integer
supporting rail assembly possible	
quantity the RJ45 Ports	
manufacturer's item number	
manufacturer's name	


```
<SystemUnitClass Name="FL Switch 5TX">
  <Attribute Name="rate of transmission" Unit="MBit/s" AttributeDataType="xs:integer">
    <DefaultValue>10</DefaultValue>
    <Value>100</Value>
  </Attribute>
  <Attribute Name="supporting rail assembly possible" AttributeDataType="xs:boolean">
    <Value>true</Value>
  </Attribute>
  <Attribute Name="quantity the RJ45 Ports" AttributeDataType="xs:integer">
    <Value>5</Value>
  </Attribute>
  <Attribute Name="manufacturer's item number" AttributeDataType="xs:string">
    <Value>2832085</Value>
  </Attribute>
  <Attribute Name="manufacturer's name" AttributeDataType="xs:string">
    <Value>Phoenix Contact</Value>
  </Attribute>
  <InternalElement Name="Endpointlist" ID="{b715818b-abdf-43af-b2a1-82873f515f85}">
    <ExternalInterface Name="EthernetSocket1" ID="{a925d6aa-8a5c-49d6-aaeb-ea1642cd816e}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <ExternalInterface Name="EthernetSocket2" ID="{c0ab19a1-ad1c-4701-b830-c2cd1f305d07}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <ExternalInterface Name="EthernetSocket3" ID="{154bd19d-993e-4418-b5b7-be8965df5c56}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <ExternalInterface Name="EthernetSocket4" ID="{1744fec4-b4cf-44dc-98b1-b92d820aee87}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <ExternalInterface Name="EthernetSocket5" ID="{eee0204a-8e9f-4baf-9baf-563cc72419bc}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetSocket" />
    <RoleRequirements RefBaseRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice/PhysicalEthernetEndpointlist" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetDevice" />
</SystemUnitClass>
```


Figure 53 - System unit class for physical device example 3


Table 30 - SystemUnitClass FL Switch 5TX


Name	FL Switch 5TX	
Description	5 Port Ethernet Switch	
Attributes	Name	transfer rate
	Value	100
	Default Value	10
	Unit	Mbit/s
	DataType	xs:integer
	Name	possibility of assembling supporting rails
	Value	1
	DataType	xs:boolean
	Name	quantity of RJ45-Ports
	Value	5
	DataType	xs:integer
	Name	manufacturer's item number
	Value	2832085
	DataType	xs:string
	Name	manufacturer
	Value	Phoenix Contact
	DataType	xs:string
Endpointlist	Role	ModellAF3RoleClassLib/PhysicalEthernetEndpointlist
	EthernetSocket1-5	ModellAF3CommunicationInterfaceClassLib/

	ParentClass	EthernetSocket
SupportedRole Class	ModellIAF3RoleClassLib/PhysicalEthernetDevice	

 Ethernet Wire { **Class:** }

 EthernetPlug1 { **Class:** EthernetPlug }

 EthernetPlug2 { **Class:** EthernetPlug }

 SupportedRoleClass: PhysicalEthernetConnection

interface type	
Name	interface type
Description	indication of the connector/base
Value	RJ45
Default Value	
Unit	
DataType	xs:string

wire length	
product type indication of	

```

<SystemUnitClass Name="Ethernet Wire">
  <Attribute Name="interface type" AttributeDataType="xs:string">
    <Description>indication of the connector/base</Description>
    <Value>RJ45</Value>
  </Attribute>
  <Attribute Name="wire length" Unit="m" AttributeDataType="xs:float">
    <Value>0.2</Value>
  </Attribute>
  <Attribute Name="product type indication of" AttributeDataType="xs:string">
    <Description>Name the product family or variation the product</Description>
    <Value>CAT5E</Value>
  </Attribute>
  <ExternalInterface Name="EthernetPlug1" ID="{4995c8d8-b8a6-4afd-b63f-ef007b93968f}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetPlug" />
  <ExternalInterface Name="EthernetPlug2" ID="{3d166d34-969d-4ab0-ac15-92afd407af6e}" RefBaseClassPath="ModellAF3CommunicationInterfaceClassLib/EthernetPlug" />
  <SupportedRoleClass RefRoleClassPath="ModellAF3RoleClassLib/PhysicalEthernetConnection" />
</SystemUnitClass>

```

Figure 54 - System unit class for physical connection example

Table 31 - SystemUnitClass Ethernet Wire

Table 37: SystemMetadata Ethernet Wire		
Name	Ethernet Wire	
Description	Network Cable	
Attributes	Name	interface type
	Value	RJ45
	DataType	xs:string
	Name	Cable Length
	Value	0.2
	Unit	m
	DataType	xs:real
	Name	product type indication
	Value	CAT5E
	DataType	xs:string
EthernetPlug1-2 ParentClass	ModelIIAF3CommunicationInterfaceClassLib/ EthernetPlug	
SupportedRole Class	ModelIIAF3RoleClassLib/PhysicalEthernetConnection	

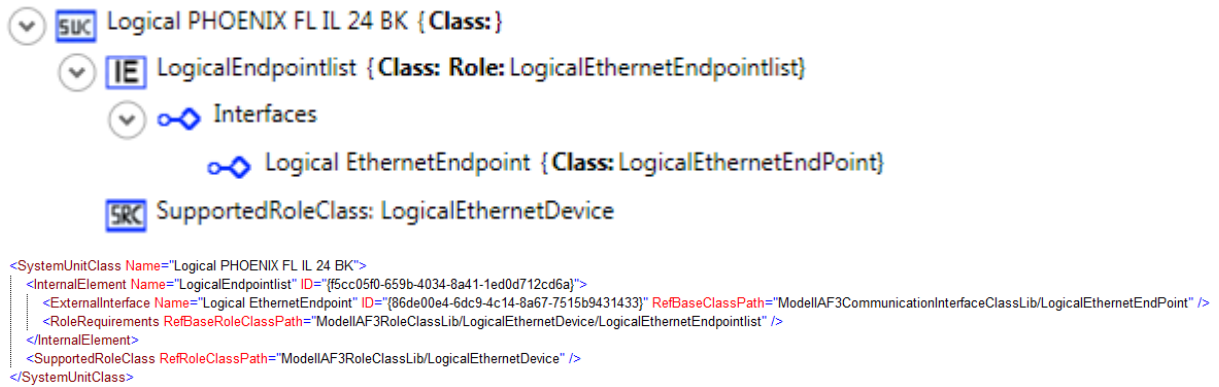


Figure 55 - System unit class for logical device example 1

Table 32 - SystemUnitClass Logical PHOENIX FL IL 24 BK

Name	Logical PHOENIX FL IL 24 BK	
Description	Bus Coupler Modbus/TCP – logical Device	
LogicalEndpointlist	Role	ModellAF3RoleClassLib/LogicalEthernetEndpointlist
	Logical EthernetEndpoint ParentClass	ModellAF3CommunicationInterfaceClassLib/LogicalEthernetEndPoint
SupportedRole Class	ModellAF3RoleClassLib/LogicalEthernetDevice	

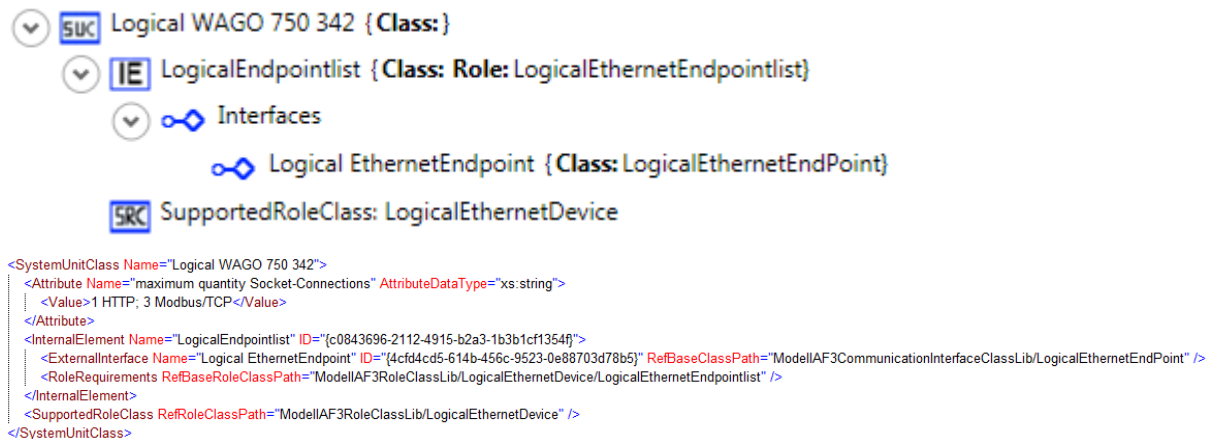


Figure 56 - System unit class for logical device example 2

Table 33 - SystemUnitClass Logical WAGO 750 342

Name	Logical WAGO 750 342	
Description	Bus Coupler Modbus/TCP – logical Device	
Attributes	Name	maximal quantity of socket connections
	Value	1 HTTP; 3 Modbus/TCP
	Data Type	xs:String
LogicalEndpointlist	Role	ModellAF3RoleClassLib/LogicalEthernetEndpointlist
	Logical	ModellAF3CommunicationInterfaceClassLib/

	EthernetEndpoint ParentClass	LogicalEthernetEndpoint
SupportedRole Class	ModellIAF3RoleClassLib/LogicalEthernetDevice	

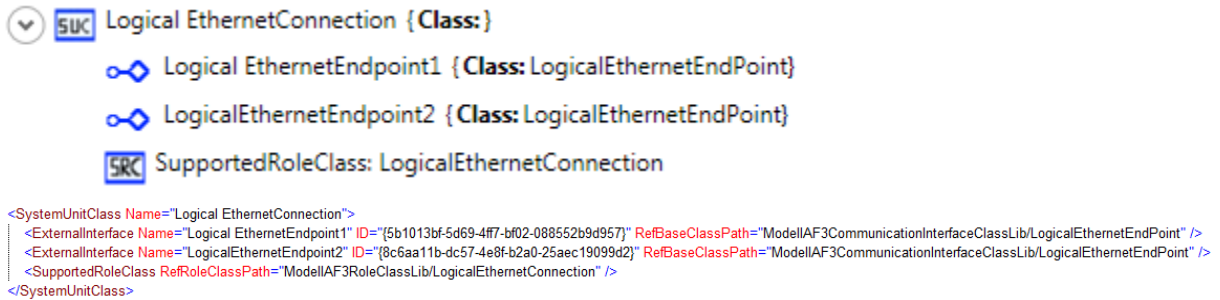


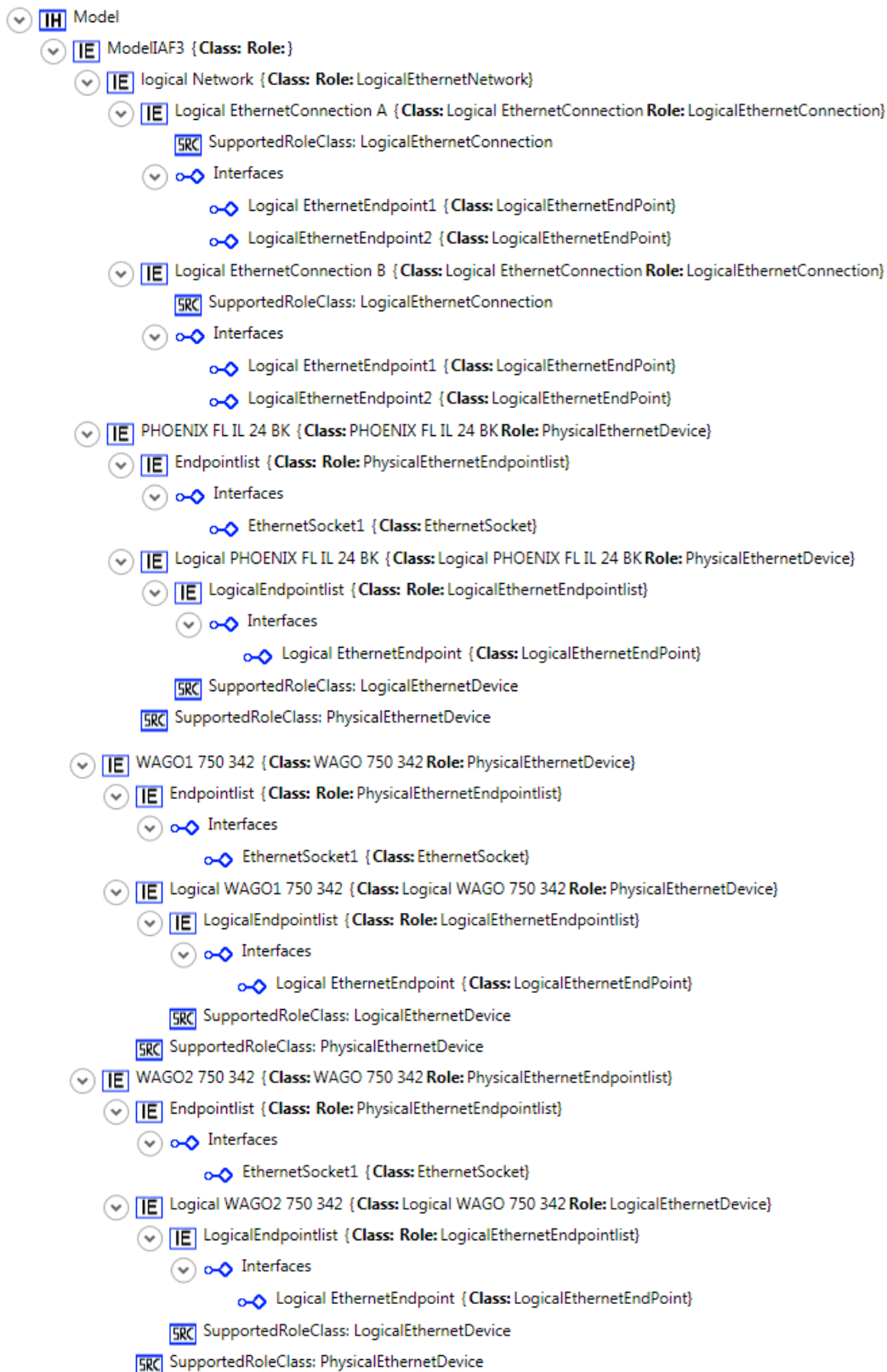
Figure 57 - System unit class for logical connection example

Table 34 - SystemUnitClass LogicalEthernetConnection

Name	LogicalEthernetConnection
Description	logical connection, Ethernet Cable on the Basis of Logic
LogicalEthernet Endpoint1-2 ParentClass	ModellIAF3CommunicationInterfaceClassLib/ LogicalEthernetEndPoint
SupportedRole Class	ModellIAF3RoleClassLib/LogicalEthernetDevice

6.1.4 Instance Hierarchy

Exploiting the defined system unit classes the following instance hierarchy modelling the lab size manufacturing system communication system example can be developed.



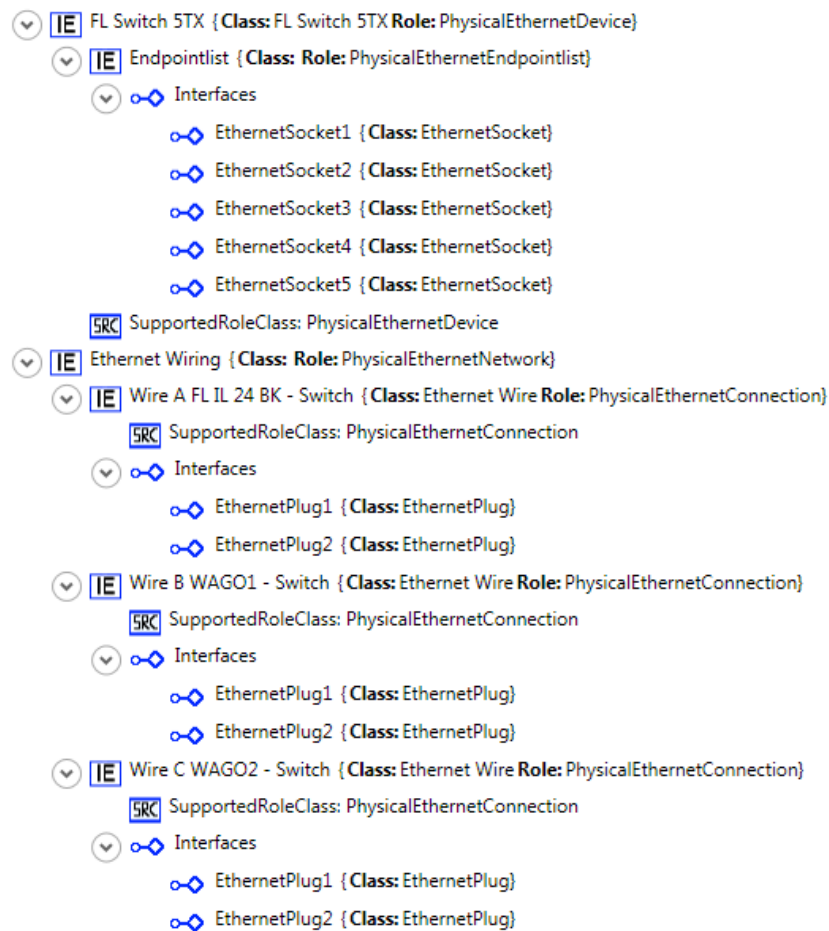
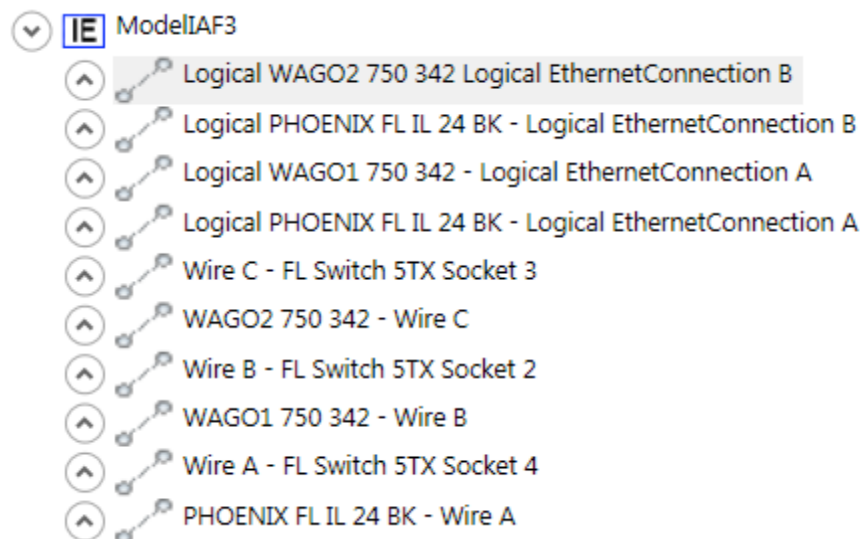


Figure 58 - Instance hierarchy for communication system example

6.1.5 Links

To link the related elements of the instance hierarchy given in section 6.1.4 the following internal links can be used.



```

<InternalLink Name="Logical WAGO2 750 342 - Logical EthernetConnection B" RefPartnerSideA="f1aaf97aa-6f19-4812-ac72-064678dd6444" Logical EthernetEndpoint" RefPartnerSideB="352a61ce-4ede-4c5b-a6cb-0d9486e0d94b" Logical EthernetEndpoint2" />
<InternalLink Name="Logical PHOENIX FL IL 24 BK - Logical EthernetConnection B" RefPartnerSideA="519dd567-ca9a-40af-aa95-91974293c9e4" Logical EthernetEndpoint" RefPartnerSideB="352a61ce-4ede-4c5b-a6cb-0d9486e0d94b" Logical EthernetEndpoint1" />
<InternalLink Name="Logical WAGO1 750 342 - Logical EthernetConnection A" RefPartnerSideA="1a495b58-1cea-4cbb-af3-4440306099e" Logical EthernetEndpoint" RefPartnerSideB="01a76005-7837-47c2-be28-14a8a1d3e452" Logical EthernetEndpoint2" />
<InternalLink Name="Logical PHOENIX FL IL 24 BK - Logical EthernetConnection A" RefPartnerSideA="01a76005-7837-47c2-be28-14a8a1d3e452" Logical EthernetEndpoint1" RefPartnerSideB="519dd567-ca9a-40af-aa95-91974293c9e4" Logical EthernetEndpoint" />
<InternalLink Name="Wire C - FL Switch STX Socket 3" RefPartnerSideA="ad2944e9-2573-4331-b8a9-d63af3ca20" EthernetPlug2" RefPartnerSideB="be9049cc-32e8-44b8-a769-0d55947e037f" EthernetSocket3" />
<InternalLink Name="WAGO2 750 342 - Wire C" RefPartnerSideA="1366a9e97-820b-49e5-9d24-6d43846a133b" EthernetSocket1" RefPartnerSideB="ad2944e9-2573-4331-b8a9-d63af3ca20" EthernetPlug1" />
<InternalLink Name="Wire B - FL Switch STX Socket 2" RefPartnerSideA="c8d61340-c65b-4594-8c0a-c18d5b697b90" EthernetPlug2" RefPartnerSideB="be9049cc-32e8-44b8-a769-0d55947e037f" EthernetSocket2" />
<InternalLink Name="WAGO1 750 342 - Wire B" RefPartnerSideA="a5828e7-d47b-4808-a604-3c47293fd43" EthernetSocket1" RefPartnerSideB="c8d61340-c65b-4594-8c0a-c18d5b697b90" EthernetPlug1" />
<InternalLink Name="Wire A - FL Switch STX Socket 4" RefPartnerSideA="9744197c-0de0-462f-bc32-6ce5e4f564" EthernetPlug2" RefPartnerSideB="be9049cc-32e8-44b8-a769-0d55947e037f" EthernetSocket4" />
<InternalLink Name="PHOENIX FL IL 24 BK - Wire A" RefPartnerSideA="b7964463-ba1c-4ae7-97e1-fd88e5db3353" EthernetSocket1" RefPartnerSideB="9744197c-0de0-462f-bc32-6ce5e4f564" EthernetPlug1" />

```

Figure 59 - Internal links related to the instance hierarchy for communication system example

6.2 PROFINET demonstrator

The second example shall demonstrate the usage of AutomationML in order to describe a small productions system consisting of a PLC, an input device and an output device. This example will put the focus on modelling of communication data packages (PDUs) (see Figure 60).

The input device provides 2 data of type Boolean consumed by the main controller while the output device consumes 1 Boolean parameter calculated and provided by the main controller based on the input signals.

These are the three Boolean parameters D1, D2 and D3 that are mapped inside the PLC (the program is not detailed described here). The variables D1 and D2 are mapped to PN_1_1_IN00 and PN_1_1_IN03 as input ports. The PLC calculates output variable D3 which is mapped to PN_1_2_OUT01. These values will be sent cyclically to the output device as Boolean parameters too.

The communication protocol used in this example is PROFINET. The devices are connected in a line topology. Thus the devices offer 2 Ethernet sockets. Internally these sockets are connected by means of a switch, so datagrams will be forwarded (see Figure 61).

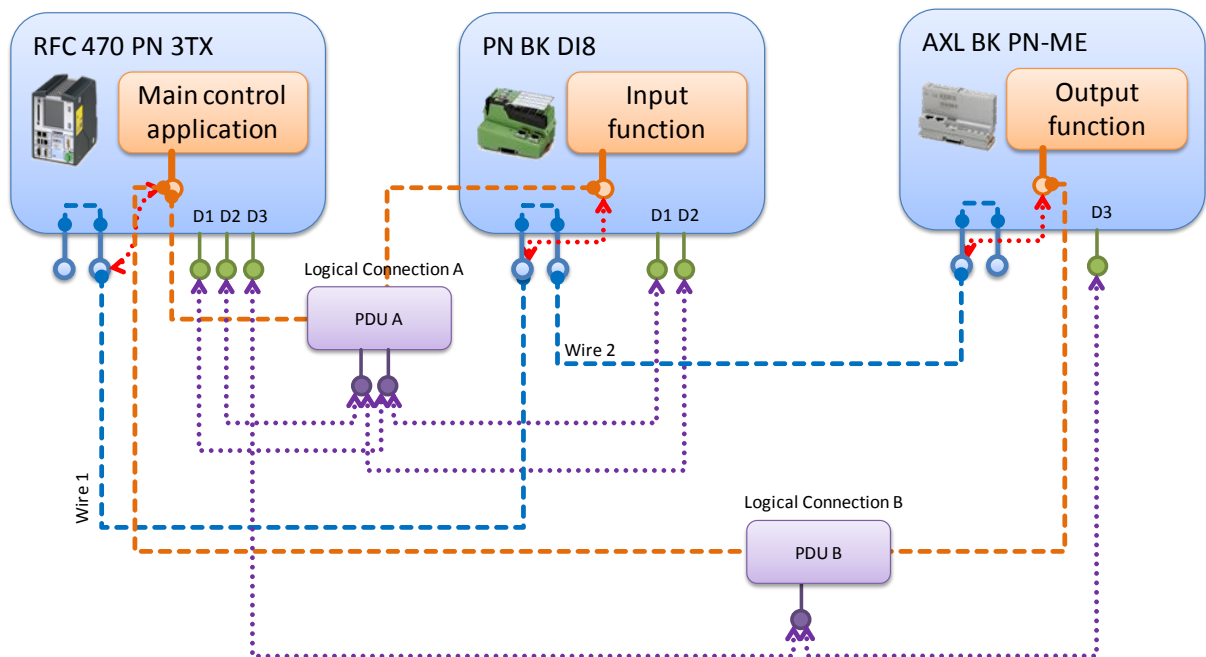
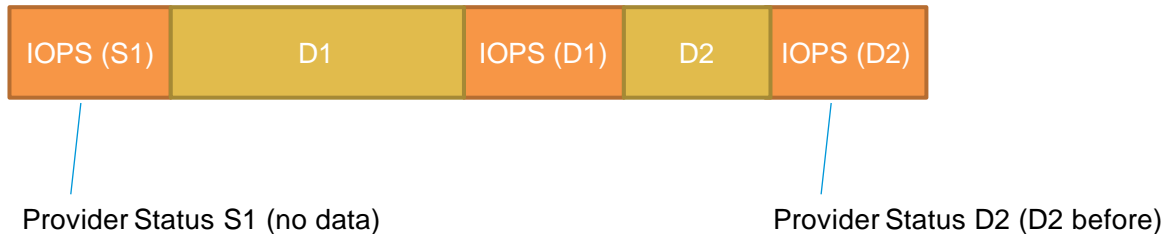


Figure 60 - Physical and logical topology of PROFINET demonstrator

Controller → Sensor



Sensor → Controller



Controller → Aktor



Aktor → Controller

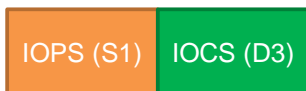
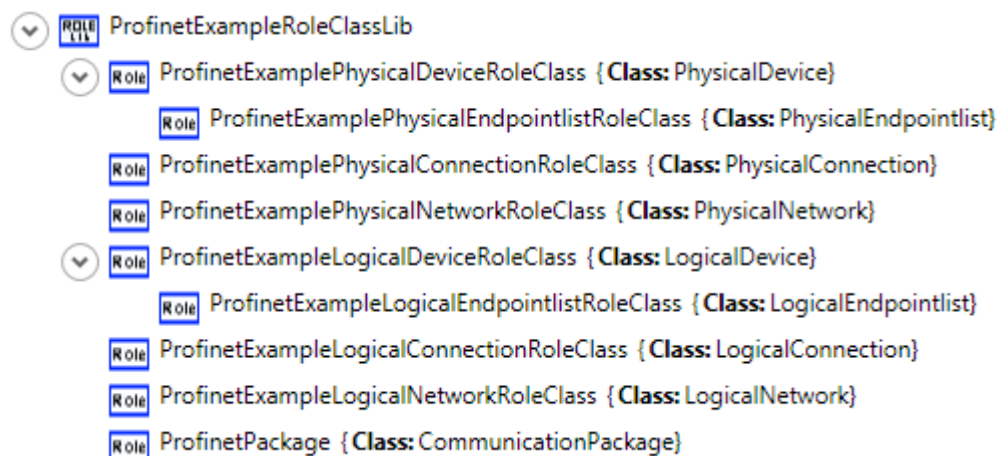


Figure 61 - Communication packages of PROFINET demonstrator

The modelling steps presented in Section 5.5 are used to model the following artefacts of the communication system. The complete *.aml file is given on the web page of AutomationML under <https://www.automationml.org/o.red.c/dateien.html>.

6.2.1 Role classes

The following role class library has been derived following sections 5.4.2 and 5.5.3.2 from the basic roles classes defined in section 5.2 and the extended role classes defined in section 5.5.



```

<RoleClassLib Name="ProfinetExampleRoleClassLib">
  <Description>AutomationML Communication Role Class Library Example Profinet</Description>
  <Version>0.9</Version>
  <RoleClass Name="ProfinetExamplePhysicalDeviceRoleClass" RefBaseClassPath="CommunicationRoleClassLib/PhysicalDevice">
    <RoleClass Name="ProfinetExamplePhysicalEndpointlistRoleClass" RefBaseClassPath="CommunicationRoleClassLib/PhysicalDevice/PhysicalEndpointlist" />
  </RoleClass>
  <RoleClass Name="ProfinetExamplePhysicalConnectionRoleClass" RefBaseClassPath="CommunicationRoleClassLib/PhysicalConnection" />
  <RoleClass Name="ProfinetExamplePhysicalNetworkRoleClass" RefBaseClassPath="CommunicationRoleClassLib/PhysicalNetwork" />
  <RoleClass Name="ProfinetExampleLogicalDeviceRoleClass" RefBaseClassPath="CommunicationRoleClassLib/LogicalDevice">
    <RoleClass Name="ProfinetExampleLogicalEndpointlistRoleClass" RefBaseClassPath="CommunicationRoleClassLib/LogicalDevice/LogicalEndpointlist" />
  </RoleClass>
  <RoleClass Name="ProfinetExampleLogicalConnectionRoleClass" RefBaseClassPath="CommunicationRoleClassLib/LogicalConnection" />
  <RoleClass Name="ProfinetExampleLogicalNetworkRoleClass" RefBaseClassPath="CommunicationRoleClassLib/LogicalNetwork" />
  <RoleClass Name="ProfinetPackage" RefBaseClassPath="CommunicationRoleClassLib/CommunicationPackage">
  </RoleClass>
</RoleClassLib>

```

Figure 62 - Role class library for PROFINET demonstrator example

Table 35 - Role ProfinetExamplePhysicalConnectionRoleClass

Name	ProfinetExamplePhysicalConnectionRoleClass
Description	Physical Network Connections specifically for PROFINET Networks
ParentClass	CommunicationRoleClassLib/PhysicalConnection

Table 36 - Role ProfinetExamplePhysicalDeviceRoleClass

Name	ProfinetExamplePhysicalDeviceRoleClass
Description	Physical Network Devices specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/PhysicalDevice

Table 37 - Role ProfinetExamplePhysicalEndpointlistRoleClass

Name	ProfinetExamplePhysicalEndpointlistRoleClass
Description	Physical Network Devices Port List specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/PhysicalEndpointlist

Table 38 - Role ProfinetExamplePhysicalNetworkRoleClass

Name	ProfinetExamplePhysicalNetworkRoleClass
Description	Physical Network specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/PhysicalNetwork

Table 39 - Role ProfinetExampleLogicalConnectionRoleClass

Name	ProfinetExampleLogicalConnectionRoleClass
Description	Logical Network Connection specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/LogicalConnection

Table 40 - Role ProfinetExampleLogicalDeviceRoleClass

Name	ProfinetExampleLogicalDeviceRoleClass
Description	Logical Network Device specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/LogicalDevice

Table 41 - Role ProfinetExampleLogicalEndpointlistRoleClass

Name	ProfinetExampleLogicalEndpointlistRoleClass
Description	Logical Network Device Port List specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/LogicalEndpointlist

Table 42 - Role ProfinetExampleLogicalNetworkRoleClass

Name	ProfinetExampleLogicalNetworkRoleClass
Description	Logisches Network specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/LogicalNetwork

Table 43 - Role ProfinetPackage

Name	ProfinetPackage
Description	Profinet Package specifically for PROFINET Network
ParentClass	CommunicationRoleClassLib/CommunicationPackage

6.2.2 Interface classes

The following interface class library has been derived following sections 5.4.3 and 5.5.3.3 from the basic interface classes defined in section 5.2. and the extended interface classes defined in section 5.5.

```
<InterfaceClassLib Name="ProfinetExampleInterfaceClassLib">
  <Description>AutomationML Communication Interface Class Library Example Profinet</Description>
  <Version>0.9</Version>
  <InterfaceClass Name="ProfinetExamplePhysicalPlug" RefBaseClassPath="CommunicationInterfaceClassLib/PhysicalEndPoint" />
  <InterfaceClass Name="ProfinetExamplePhysicalSocket" RefBaseClassPath="CommunicationInterfaceClassLib/PhysicalEndPoint" />
  <InterfaceClass Name="ProfinetExampleLogicalEndPointInterfaceClass" RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <InterfaceClass Name="ProfinetDatagrammObject" RefBaseClassPath="CommunicationInterfaceClassLib/DatagrammObject" ID="2a4c28db-f5a5-4341-a99e-2120bcc6708b" />
</InterfaceClassLib>
```

Figure 63 - Interface class library for PROFINET demonstrator example

Table 44 - InterfaceClass ProfinetExamplePhysicalPlug

Name	ProfinetExamplePhysicalPlug
Description	Physical Network Plug specifically for physical PROFINET Networks
ParentClass	CommunicationInterfaceClassLib/PhysicalEndPoint

Table 45 - InterfaceClass ProfinetExamplePhysicalSocket

Name	ProfinetExamplePhysicalSocket
Description	Physical Network Socket specifically for physical PROFINET Networks
ParentClass	CommunicationInterfaceClassLib/PhysicalEndPoint

Table 46 - InterfaceClass ProfinetExampleLogicalEndPointInterfaceClass

Name	ProfinetExampleLogicalEndPointInterfaceClass
Description	Terminal Points specifically for logical PROFINET Network
ParentClass	AutomationMLInterfaceClassLib/AutomationMLBaseInterface

Table 47 - InterfaceClass ProfinetProfinetDatagrammObject

Name	ProfinetProfinetDatagrammObject
Description	PDU content element within a PROFINET network
ParentClass	CommunicationInterfaceClassLib/DatagrammObject

6.2.3 System Unit Classes

The following system unit class libraries have been derived following sections 5.4.4 and 5.5.3.4.

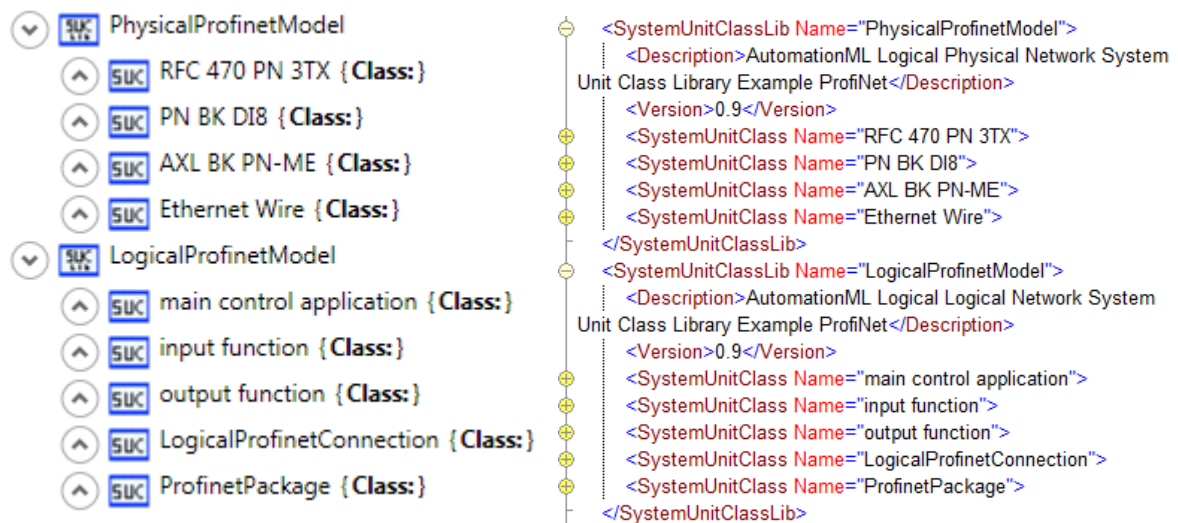


Figure 64 - System unit class library for PROFINET demonstrator example

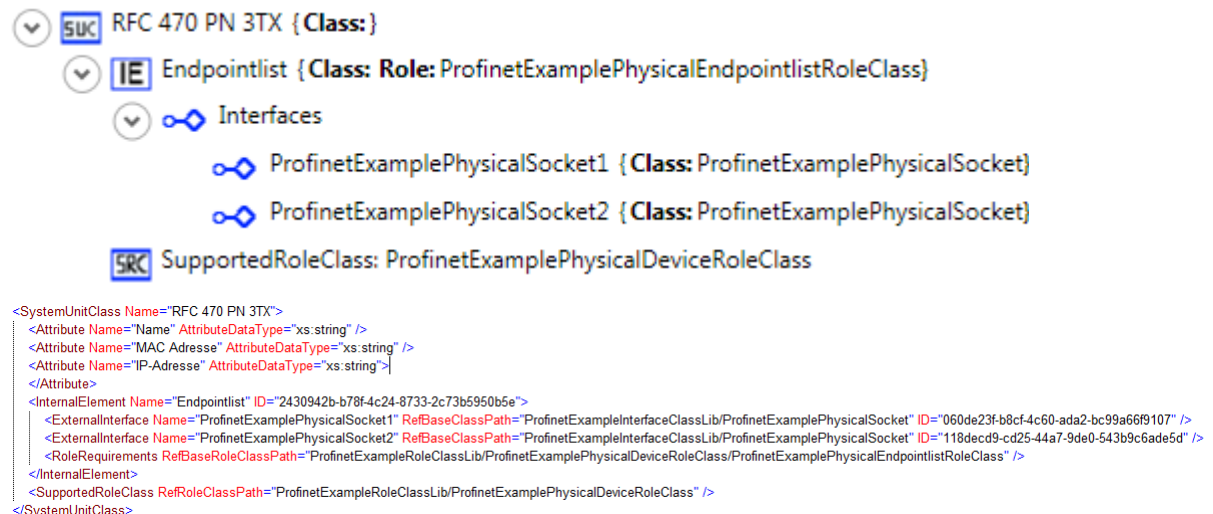


Figure 65 - System unit class for physical PROFINET device example 1

Table 48 - SystemUnitClass RFC 470 PN 3TX

Name	RFC 470 PN 3TX	
Description	Main controller hardware device	
Attributes	Name	Name
	DataType	xs:string
	Name	MAC Adresse
	DataType	xs:string
	Name	IP-Adresse
	DataType	xs:string
Endpointlist	Role	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalEndpointlistRoleClass
	ProfinetExamplePhysicalSocket1	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket
	ProfinetExamplePhysicalSocket2	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket
SupportedRoleClass	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass	

PN BK DI8 {Class:}

Endpointlist {Class: Role: ProfinetExamplePhysicalEndpointlistRoleClass}

Interfaces

ProfinetExamplePhysicalSocket1 {Class: ProfinetExamplePhysicalSocket}

ProfinetExamplePhysicalSocket2 {Class: ProfinetExamplePhysicalSocket}

SupportedRoleClass: ProfinetExamplePhysicalDeviceRoleClass

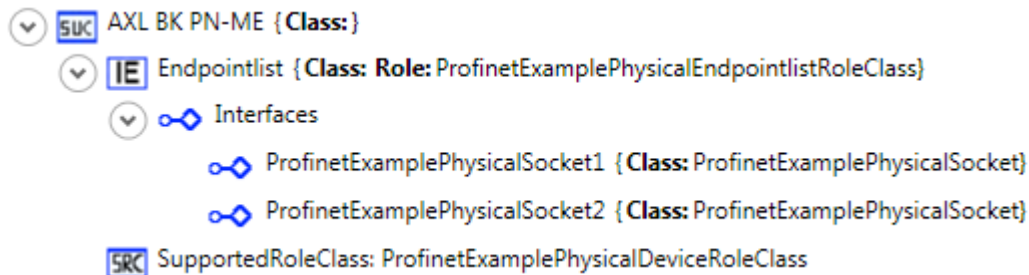
```
<SystemUnitClass Name="PN BK DI8">
  <Attribute Name="Name" AttributeDataType="xs:string" />
  <Attribute Name="MAC Adresse" AttributeDataType="xs:string" />
  <Attribute Name="IP-Adresse" AttributeDataType="xs:string" />
  <InternalElement Name="Endpointlist" ID="2fc69310-e6d0-482c-802b-0bd69f57dcb3">
    <ExternalInterface Name="ProfinetExamplePhysicalSocket1" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket" ID="f40ee17f-5b00-4fdb-96c1-d2b8e5a869c6" />
    <ExternalInterface Name="ProfinetExamplePhysicalSocket2" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket" ID="9153d942-a2c8-496e-bd27-b2c4ab0242a6" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass" />
</SystemUnitClass>
```

Figure 66 - System unit class for physical PROFINET device example 2

Table 49 - SystemUnitClass PN BK DI8

Name	PN BK DI8	
Description	IO device / control input bus coupler	
Attributes	Name	Name
	DataType	xs:string
	Name	MAC Adresse
	DataType	xs:string

Endpointlist	Name	IP-Adresse
	DataType	xs:string
	Role	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalEndpointlistRoleClass
	ProfinetExamplePhysicalSocket1	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket
SupportedRoleClass	ProfinetExamplePhysicalSocket2	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket
	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass	



```

<SystemUnitClass Name="AXL BK PN-ME">
  <Attribute Name="Name" AttributeDataType="xs:string" />
  <Attribute Name="MAC Adresse" AttributeDataType="xs:string" />
  <Attribute Name="IP-Adresse" AttributeDataType="xs:string" />
  <InternalElement Name="Endpointlist" ID="b4a8d8b1-9f18-4315-9782-34d9c9905e13">
    <ExternalInterface Name="ProfinetExamplePhysicalSocket1" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket" ID="95d01e9a-8317-448e-bc4f-1fbb3d54d07f" />
    <ExternalInterface Name="ProfinetExamplePhysicalSocket2" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket" ID="4d02a925-9cd3-44ba-83a4-66befabf6527" />
    <RoleRequirements RefBaseRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass/ProfinetExamplePhysicalEndpointlistRoleClass" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass" />
</SystemUnitClass>


```

Figure 67 - System unit class for physical PROFINET device example 3

Table 50 - SystemUnitClass AXL BK PN-ME


Name	AXL BK PN-ME	
Description	IO device / control output bus coupler	
Attributes	Name	Name
	DataType	xs:string
	Name	MAC Adresse
	DataType	xs:string
	Name	IP-Adresse
	DataType	xs:string
Endpointlist	Role	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalEndpointlistRoleClass
	ProfinetExamplePhysicalSocket1	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket
	ProfinetExamplePhysicalSocket2	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalSocket

SupportedRole Class	ProfinetExampleRoleClassLib/ProfinetExamplePhysicalDeviceRoleClass
----------------------------	--

▼  Ethernet Wire { Class: }

ProfinetExamplePhysicalPlug 1 { Class: ProfinetExamplePhysicalPlug }

ProfinetExamplePhysicalPlug 2 { Class: ProfinetExamplePhysicalPlug }


 SupportedRoleClass: ProfinetExamplePhysicalConnectionRoleClass


```
<SystemUnitClass Name="Ethernet Wire">
  <ExternalInterface Name="ProfinetExamplePhysical Plug 1" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalPlug" ID="0681ae91-fb3a-42aa-8015-f6f93307e63b" />
  <ExternalInterface Name="ProfinetExamplePhysical Plug 2" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalPlug" ID="ba50eb61-15f8-4466-a90f-5125083caed8" />
  <SupportedRoleClass RefRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExamplePhysicalConnectionRoleClass" />
</SystemUnitClass>
```


Figure 68 - System unit class for physical PROFINET connection example


Table 51 - SystemUnitClass Ethernet Wire


Name	Ethernet Wire
Description	Network Cable for PROFINET wiring
EthernetPlug1-2 ParentClass	ProfinetExampleInterfaceClassLib/ProfinetExamplePhysicalPlug
SupportedRole Class	ProfinetExampleRoleClassLib/ ProfinetExamplePhysicalConnectionRoleClass


▼  main control application { Class: }


▼  LogicalEndpointlist { Class: Role: ProfinetExampleLogicalDeviceRoleClass }


▼  Interfaces

 D1 { Class: VariableInterface }

 D2 { Class: VariableInterface }

 D3 { Class: VariableInterface }

 ProfinetExampleLogicalEndPointInterfaceClass { Class: ProfinetExampleLogicalEndPointInterfaceClass }

 SupportedRoleClass: ProfinetExampleLogicalDeviceRoleClass

```
<SystemUnitClass Name="main control application">
  <InternalElement Name="LogicalEndpointlist" ID="b73a6d82-b07e-4d75-a64f-4e07d7f50165">
    <ExternalInterface Name="D1" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="0dedec0e-fa7d-48bb-aad0-5436b3c6bd68">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" />
    </ExternalInterface>
    <ExternalInterface Name="D2" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="183e86c3-934b-4043-8401-16048c60d5ea">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" />
    </ExternalInterface>
    <ExternalInterface Name="D3" RefBaseClassPath="AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface" ID="62614add-cc16-46d6-8491-45c61cee3bdc3">
      <Attribute Name="refURI" AttributeDataType="xs:anyURI" />
    </ExternalInterface>
    <ExternalInterface Name="ProfinetExampleLogicalEndPointInterfaceClass" RefBaseClassPath="ProfinetExampleInterfaceClassLib/ProfinetExampleLogicalEndPointInterfaceClass" ID="25626d29-7253-4187-9d6e-819cee2bd8a">
      <RoleRequirements RefBaseRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExampleLogicalDeviceRoleClass" />
    </ExternalInterface>
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath="ProfinetExampleRoleClassLib/ProfinetExampleLogicalDeviceRoleClass" />
</SystemUnitClass>
```

Figure 69 - System unit class for logical PROFINET device example 1

Table 52 - SystemUnitClass main control application

Name	main control application	
Description	Main control function calculating the actor actions based on sensor signals	
LogicalEndpointlist	Role	ProfinetExampleRoleClassLib/ProfinetExampleLogicalEndpointlistRoleClass
	ProfinetExampleLogi	ProfinetExampleInterfaceClassLib/ProfinetExampleL

	calEndPointInterface Class	logicalEndPointInterfaceClass
	D1	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
	D2	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
	D3	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
SupportedRole Class	ProfinetExampleRoleClassLib/ProfinetExampleLogicalDeviceRoleClass	

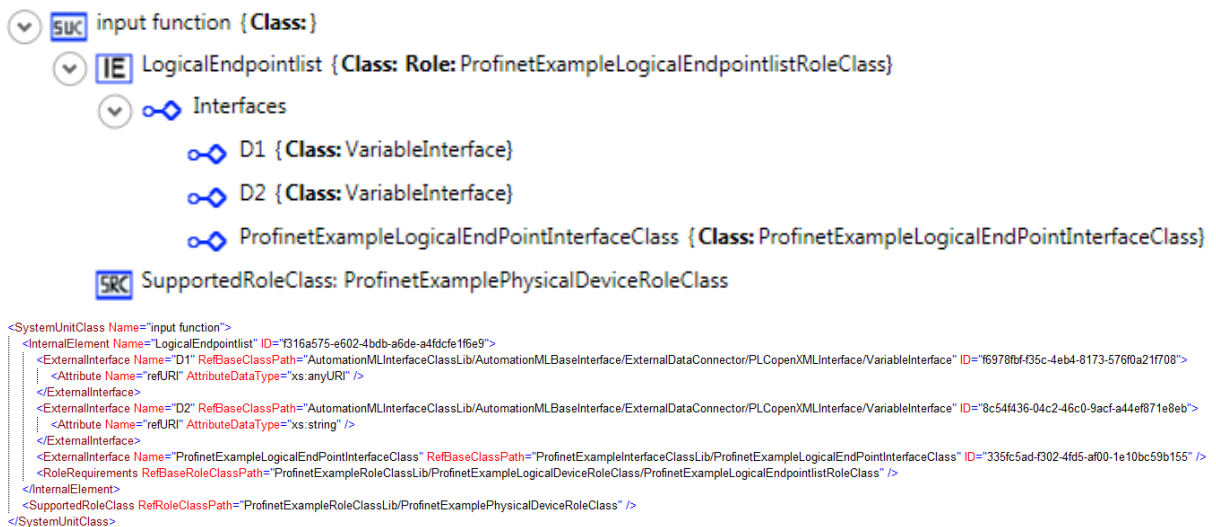


Figure 70 - System unit class for logical PROFINET device example 2

Table 53 - SystemUnitClass Input function

Name	Input function	
Description	IO function providing access to sensor signals	
LogicalEndpointlist	Role	ProfinetExampleRoleClassLib/ProfinetExampleLogicalEndpointlistRoleClass
	ProfinetExampleLogicalEndPointInterface Class	ProfinetExampleInterfaceClassLib/ProfinetExampleLogicalEndPointInterfaceClass
	D1	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
	D2	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
SupportedRole Class	ProfinetExampleRoleClassLib/ProfinetExampleLogicalDeviceRoleClass	

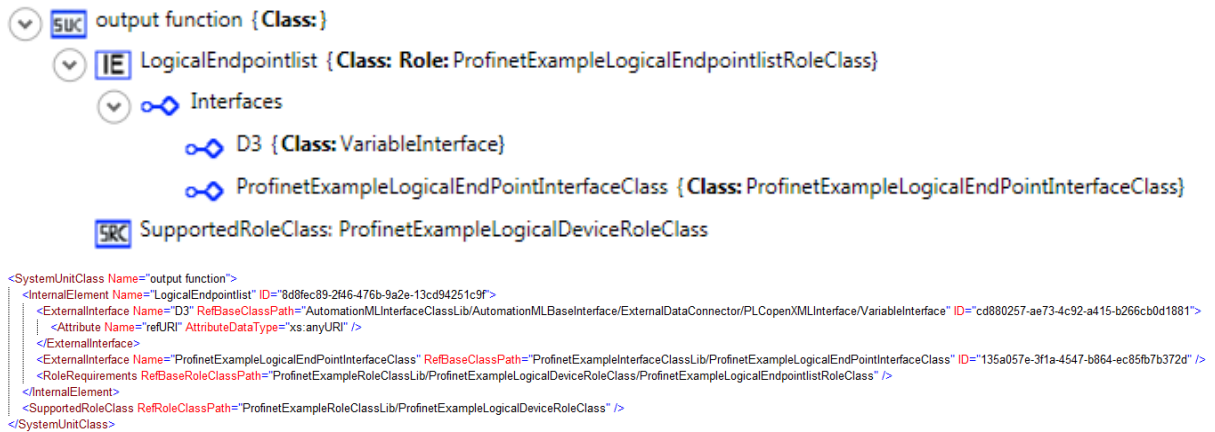


Figure 71 - System unit class for logical PROFINET device example 3

Table 54 - SystemUnitClass Output function

Name	Output function	
Description	IO function providing access to actor signals	
LogicalEndpointlist	Role	ProfinetExampleRoleClassLib/ProfinetExampleLogicalEndpointlistRoleClass
	ProfinetExampleLogicalEndPointInterfaceClass	ProfinetExampleInterfaceClassLib/ProfinetExampleLogicalEndPointInterfaceClass
	D3	AutomationMLInterfaceClassLib/AutomationMLBaseInterface/ExternalDataConnector/PLCopenXMLInterface/VariableInterface
SupportedRoleClass	ProfinetExampleRoleClassLib/ProfinetExampleLogicalDeviceRoleClass	

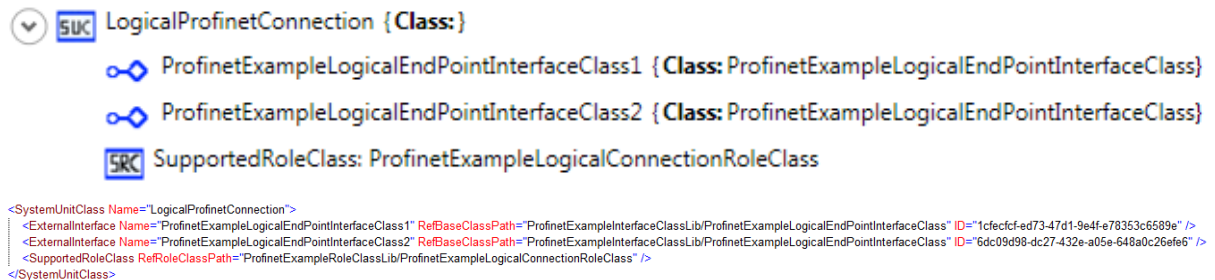





Figure 72 - System unit class for logical PROFINET connection example

Table 55 - SystemUnitClass LogicalProfinetConnection

Name	LogicalProfinetConnection	
Description	Logical connection between PROFINET logical devices	
ProfinetExampleLogicalEndPointInterfaceClass1-2 ParentClass	ProfinetExampleInterfaceClassLib/ProfinetExampleLogicalEndPointInterfaceClass	
SupportedRole	ProfinetExampleRoleClassLib/ProfinetExampleLogicalConnectionRoleClass	

Class


 ProfinetPackage {Class:}
 SupportedRoleClass: ProfinetPackage

```

<SystemUnitClass Name="ProfinetPackage">
.....
  <SupportedRoleClass RefRoleClassPath="ProfinetExampleRoleClassLib/ProfiNetPackage" />
</SystemUnitClass>
  
```

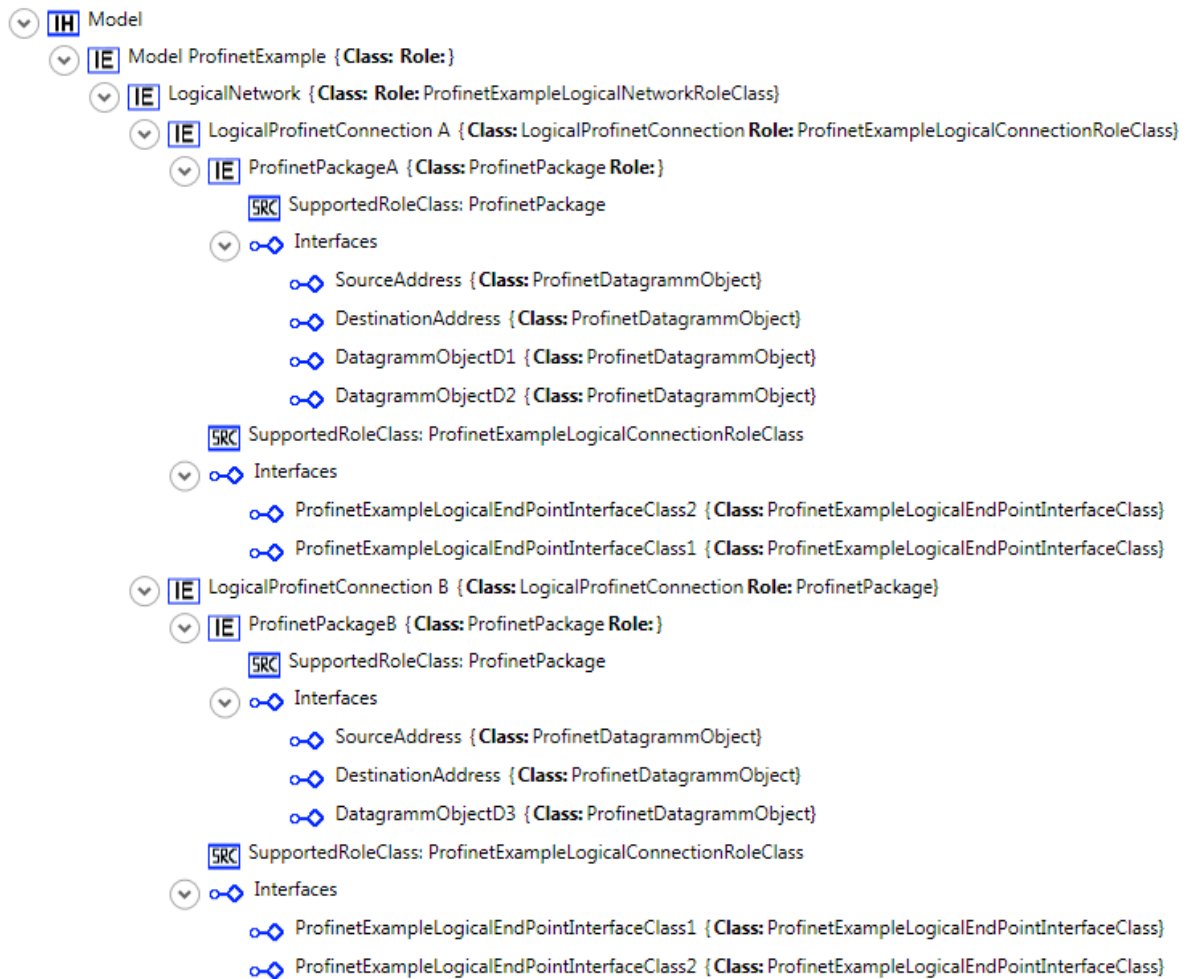
Figure 73 - System unit class for PROFINET communication package example

Table 56 - SystemUnitClass ProfinetPackage

Name	ProfinetPackage
Description	PROFINET PDU transmitted via logical connection between PROFINET logical devices
SupportedRole Class	ProfinetExampleRoleClassLib/ProfinetPackage

6.2.4 Instance Hierarchy

Exploiting the defined system unit classes the following instance hierarchy modelling the PROFINET demonstrator example can be developed.



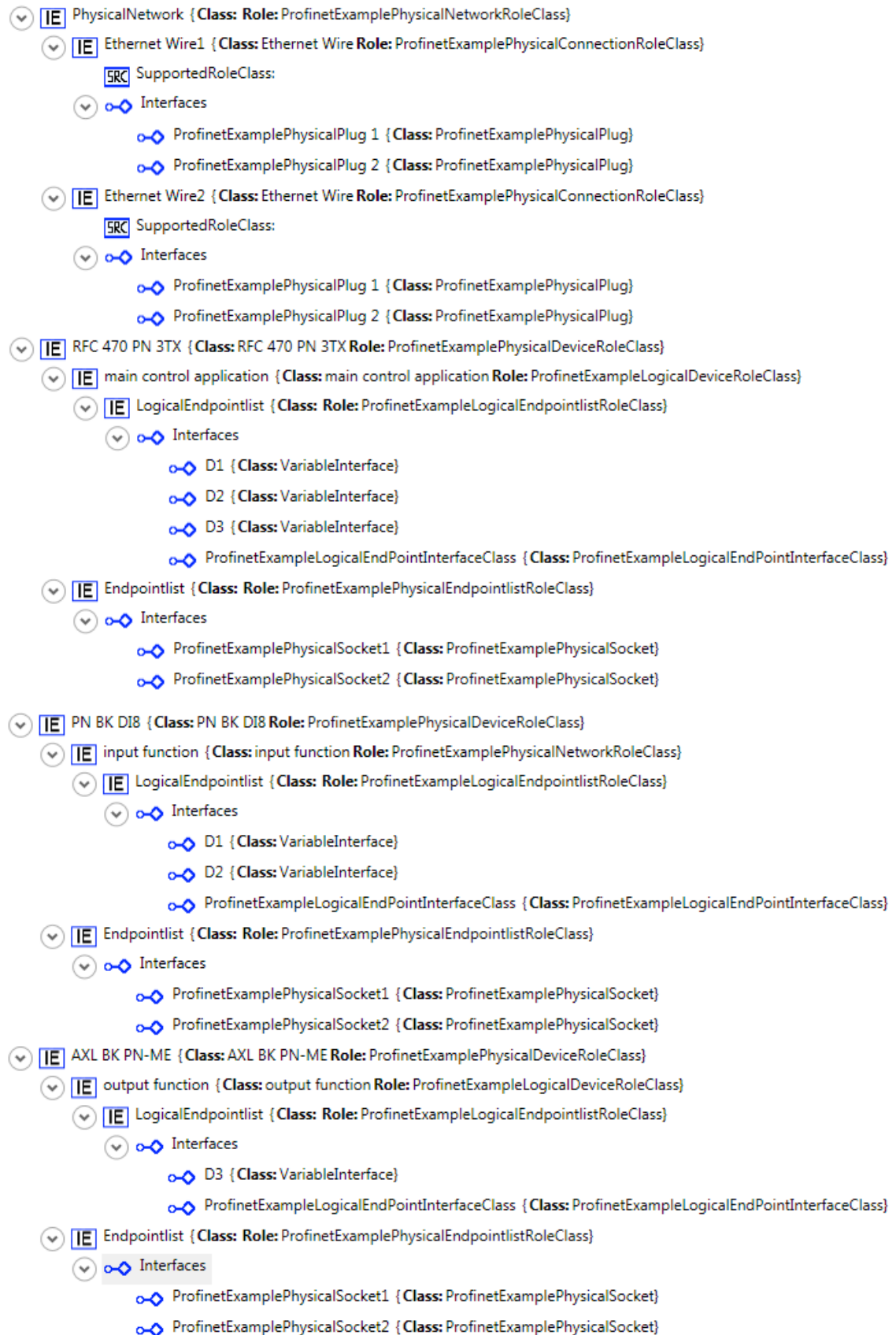


Figure 74 - Instance hierarchy for PROFINET demonstrator example

It should be noted that in this example the communication data package content is based on variables of the involved devices modelled by PLCOpenXMLInterfaces. In a similar way also SignalInterfaces can be exploited to model PDU content.

6.2.5 Links

To link the related elements of the instance hierarchy given in section 6.2.4 the following internal links can be used.



Figure 75 - Internal links related to the instance hierarchy for PROFINET demonstrator example

7 Bibliography

- [1] Object Management Group: UML Class Diagram, <http://www.uml.org/>, 2012
- [2] H. Zimmermann: OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications, Volume 28, Issue 4, pp. 425-432.
- [3] H. Kagermann, W. Wahlster, J. Helbig (Editoren): Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0 – Deutschlands Zukunft als Industriestandort sichern, Forschungsunion Wirtschaft und Wissenschaft, Arbeitskreis Industrie 4.0, [http://www.plattform-i40.de/sites/default/files/Umsetzungsempfehlungen%20 Industrie 4.0_0. pdf](http://www.plattform-i40.de/sites/default/files/Umsetzungsempfehlungen%20Industrie%204.0_0.pdf), last access Nov. 2013.
- [4] IEC 62714 (all parts), Engineering data exchange format for use in industrial systems engineering – AutomationML