Towards the Modelling of Complex Communication Networks in AutomationML

Florian Patzer*, Aranya Sarkar†, Pascal Birnstill*, Miriam Schleipen* and Jürgen Beyerer* *Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Karlsruhe, Germany
Email: {florian.patzer | pascal.birnstill | miriam.schleipen | juergen.beyerer }@iosb.fraunhofer.de †Otto von Guericke University Magdeburg, Magdeburg, Germany
Email: aranya.sarkar@st.ovgu.de

***Abstract*—For several decades production systems were considered as closed and decoupled units, where information and network security has not been an issue. This is changing rapidly, since in the age of smart factories, production systems and office IT are growing together so as to transform entire value chains into interconnected distributed systems. By this means, production systems inherit the security challenges of office IT networks connected over the Internet. Therefore and as they tend to be operated for a much longer period of time, a prospective design of security mechanisms is mandatory. For some time, the design process of production systems gets modernised by the Automation Markup Language (AutomationML, IEC 62714). AutomationML incorporates formats of all engineering phases of production systems, thus allowing engineers to model production systems on various levels of abstraction. The language also provides building blocks for modelling the network infrastructure, which are presented in the AutomationML Communication whitepaper. However, the level of detail that can be captured is currently not sufficient for modelling most network protocols and therefore any network security concept. Therefore, we propose an extension to the AutomationML Communication whitepaper and its best practice recommendations, which allows us to model networks according to the established ISO/OSI model. Using this extension we show that concepts like network separation can be modelled and validated.**

I. Introduction

The so-called fourth industrial revolution (Industry 4.0) is characterized by globally distributed and highly flexible value and supply chains. These are orchestrated beyond enterprise networks, using the Internet in order to enable new higher-level services and business models. This evolution is accompanied by the application of Internet of Things and Services (IoTS) paradigms to production systems and assets. The result is a mash-up of office and production IT networks.

Since in the past the production networks where physically isolated from external networks, they generally have no protection against cyber attacks. However, through the connection of production networks to external networks such protection is becoming a fundamental issue for those systems. As with requirements in general, also paying attention to security-related requirements becomes the more complicated and expensive the later it is dealt with. As a result, the need for considering security from the beginning of the system engineering processes arises. The corresponding paradigm is called *Security by Design (SbD)*. Network security by design requires full knowledge of the network topologies and protocols at design time which needs to be consistent with the later implemented system. Therefore, a model of the system to be designed has to be created, propagated and refined throughout all engineering phases. That way, it can be ensured that currently deployed and future security protocols as well as mechanisms are applied appropriately and security breaches are avoided. Additionally, such a maintained model allows continuous and efficient security analysis throughout the whole lifetime of the respective production networks.

To achieve such a common information base for all engineering phases and their tools has already been addressed by several approaches [1][2][3]. A promising and standardized approach is AutomationML (Automation Markup Language, further called AML) [4]. It supports all engineering phases of production systems. Unfortunately, AML lacks a methodology to describe network topologies and protocols on a level of detail such that security mechanisms can be modelled (cf. section IV). Thus, we propose an extended AML modelling concept with corresponding modelling libraries.

Our contribution extends the communication library of AML [5] by supporting the well established ISO/OSI model [6] so that we can depict networks on the granularity of its layers and according protocols (cf. section I-A). This approach can be combined with other existing modelling concepts, e.g. the application recommendation for automation project configuration [7].

This paper is structured as follows. At first, we give a short introduction to the ISO/OSI model. Afterwards, we briefly describe the aims and the structure of AML (cf. section II). In section III we introduce an example network architecture, by means of which we show the shortcomings of the current AML communication library in section IV. Based on these insights, we propose our extension of the communication library (cf. section V) and demonstrate its application using the example network architecture introduced beforehand (cf. section VI). Subsequently, we discuss related work which has been done regarding networks modelling in section VII. Finally, we discuss our contribution and open issues in section VIII.

978–1–5090–6505–9/17$31.00

A. ISO/OSI Model

The ISO/OSI model, in its latest revision [6], is the de-facto standard for describing networks and classifying protocols. It provides an abstraction for separating the different tasks of computer networking into a scheme consisting of the following seven layers:

The *physical layer* (layer 1), as the lowest layer, is concerned with physical access to a network, i.e., for transmitting the particular bits over a certain medium such as via modulating an electrical signal on a copper cable. Ethernet and its modifications, for example, define standards for such mediums on the physical layer (i.e. 1000BaseT).

The *data link layer* (layer 2) groups bits into frames and is responsible for recognizing and correcting transmission errors, which occur, for instance, due to transients on the medium. Ethernet and its modifications also operate on the data link layer. Security protocols used on the data link layer include the Extensible Authentication Protocol over LAN (EAPoL) [8], through which nodes can be authenticated before being granted access to a network.

The *network layer*, layer 3, enables communication across the boundaries of the local network, i.e., its task is to transmit data fragments from a source node to a destination node, often by routing them across several networks. The widely known protocol IP (Internet Protocol) and its security extension IPsec [9], which enables authenticated and encrypted data transmission, operate on the network layer.

The *transport layer* (layer 4) is concerned with the quality of service of end-to-end transmission of data. The Transmission Control Protocol (TCP) [10], for instance, provides a reliable service over the Internet Protocol (IP) [11], which does not cater for reliability of data transmission. The Transport Layer Security protocol (TLS) [12] enables private end-to-end communication on layer 4 using symmetric encryption and message authentication.

The *session layer* (layer 5) manages connections between communicating nodes, i.e., it is responsible for establishing, terminating, restarting, checkpointing, etc. It is usually part of an application.

The *presentation layer* (layer 6) is required once interconnected application entities use different syntax or semantics. In such cases a *presentation layer* can provide a mapping for translating between application and network data formats.

The *application layer* (layer 7), as the uppermost layer, includes the actual functionality of a network application. Security-related protocols on layer 7 include secure shell (SSH) [13] or Hypertext Transfer Protocol over TLS (HTTPS) [14].

Conceptually, the interconnection between services of one ISO/OSI layer and the one below or above is called a *Service Access Point (SAP)*, whereby each layer can only interact with its direct neighbors, i.e., the layers it shares an SAP with.

II. AutomationML

In order to reduce the complexity of highly sophisticated modern production systems, the system engineering process of the production system is broken down into various phases and this leads to different and specialized engineering tools for each phase. Naturally, a broad list of heterogeneous tools is witnessed with varied data formats and lack of support for data exchange among them [1]. Hence, AML was developed as a vendor-independent, neutral data format based on XML to support such an lossless exchange of engineering information.

The basic architecture of AML consists of the following standards:

- **CAEX**: CAEX (standardized as IEC 62424) acts as the top level format and it stores the plant topology information
- **COLLADA**: COLLADA (standardized as ISO/PAS 17506 : 2012) stores the geometric and the kinematic information.
- **PLCopen XML**: PLCopen XML is used for the storage of sequences and behavior.

AML objects (the core elements of AML) represent instances and can further consist of administration items, attributes, interfaces, relations and references [15]. AML provides the following specifications for this reason:

- **InterfaceClassLib**: The interface class library comprises of the interfaces or the description of relations among the various objects. A standard library is already existent in AML containing many abstract classes for a general automation system and these can be further extended.
- **RoleClassLib**: The role class library consists of the semantic descriptions of the AML objects, i.e. it explains the general functionality of a CAEX object within its context.
- **SystemUnitClassLib**: System unit library is the area where all the user-defined AML classes are stored for reusability. Thus, there are no concrete specifications of a certain SystemUnitClassLib in AML, however a proper guideline is defined for the design methodology.
- **InstanceHierarchy**: The concrete or the overall project data is stored in the instance hierarchies. They are the core of the AML data and are structured in a hierarchical arrangement of the object instances with their properties, interfaces, references and relations. The hierarchy is stored with nested CAEX InternalElements (IE) and each object is characterized by a unique Global Unique Identifier (GUID) and an arbitrary name [1].

Figure 1 shows these specifications in a nutshell.

The interfaces of objects are represented by the CAEX ExternalInterfaces or EIs. EIs are typically connected to each other using InternalLinks (the CAEX link mechanisms). Thus, InternalLinks model the relations between objects. To understand this concept in detail, the interested reader is referred to the 'Modelling of relations' section in [5].

III. Example Network Architecture

For better understanding of the contribution of this paper, we introduce an example of a simple network architecture using network separation via VLANs (cf. Figure 2). Since network separation is a basic security concept, such an architecture can be found in nearly every enterprise network. In section IV we

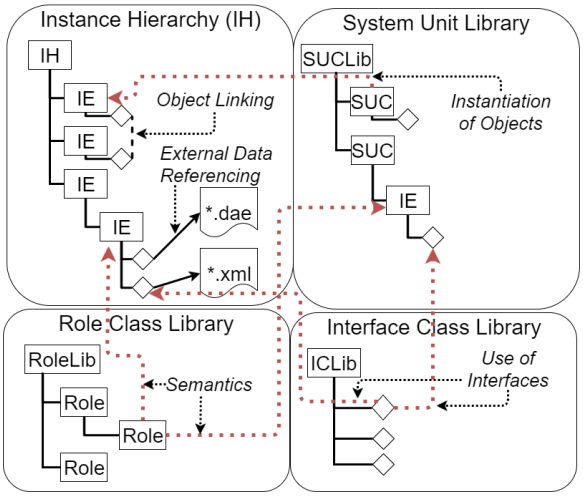

Fig. 1. Overview of the AutomationML Specifications and their relations

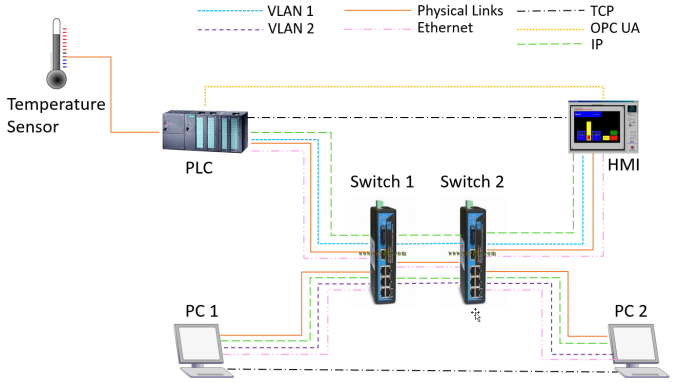

Fig. 2. Example Network Architecture

show that already this simple example cannot be sufficiently modelled using the current communication library of AML.

The architecture model consists of a temperature sensor, a PLC (Programmable Logic Controller) which hosts an OPC UA (Open Platform Communications Unified Architecture [16]) server, a HMI which serves as the OPC UA client, two engineering station PCs (PC 1 and PC 2) and two industrial switches (Switch 1 and Switch 2). A physical link based on the RS-232 standard is present between the sensor and the PLC. The PLC and PC 1 are physically connected to the RJ45 ports of Switch 1. In a similar fashion, the HMI and PC 2 are physically connected to RJ45 ports of Switch 2 using 1000BaseT as wire standard. Switch 1 and Switch 2 are also connected via a 1000BaseT wire. The protocols used for the communication between the components, namely 'Ethernet', 'IP', 'TCP' and 'OPC UA', are visualized in the architecture model as well.

Furthermore, the network is segmented using VLANs (Virtual LANs), as can been seen in Figure 2. These VLANs create isolated networks within the overall network, enhancing the network security and broadcast control (this is known as network segmentation). In our example, VLAN 1 and VLAN 2 are the tags (or names) of the two Virtual LANs, where the PLC and the HMI belong to VLAN 1 and the other devices,

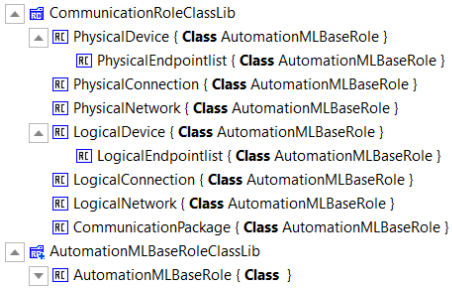

Fig. 3. Role classes of the AutomationML communication library [5]

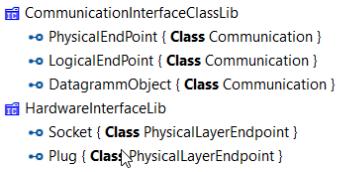

Fig. 4. Interface classes of the AutomationML communication library [5]

i.e. the PC 1 and PC 2, are assigned to VLAN 2. This is achieved by enabling a static VLAN mechanism (port-based VLANs) in the switches and assigning the correct VLAN tag to the respective port numbers of the switches.

IV. AutomationML Communication Library

The AML communication library was derived from the AutomationML Whitepaper - Communication [5]. Within [5] two types of communication related information are addressed as main targets of the modelling approach. The first type consists of communication relevant information for configuration of communication components of sensors and actuators. The second type consists of communication network configuration and structure information and includes, among others, infrastructure device configuration, wiring and quality of service information. In consequence of the gathered modelling requirements, the authors derived role and interface classes which can be used to model such communication related information.

Figures 3 and 4 show the Physical$X$ and Logical$X$ role and interface classes, where X is a placeholder for either Device, EndPoint, Connection or Network. In [5] the authors defined the Physical$X$ classes as representation of layers 1 and 2 and the Logical$X$ classes as representation of layers 3-7 of the ISO/OSI model (cf. section I-A). In the following paragraphs we briefly describe the key concepts behind those classes which are necessary to understand our contribution (cf. section V).

Physical/Logical Device: A physical device (e.g. a Switch or PLC) may contain multiple logical devices which, for example, represent applications with their own endpoints (of type LogicalEndPoint).

Physical Connection: A physical connection is modelled between EIs of devices. The interfaces have to be of the type *Socket*. The Socket class is derived from the *PhysicalEndpoint* class. Like in the real world, sockets are interfaces where one is located on every connection partner device. The counterpart

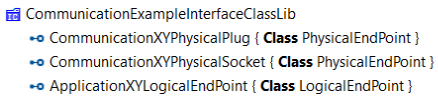

Fig. 5. Example for technology specific interface classes of the AutomationML communication library [5]

of a Socket is, again like in the real world, an EI of the type *Plug*. The Plug class is also derived from PhysicalEndpoint. Multiple implementations of transmission mediums can fulfill the role of a physical connection. Therefore, the library consists of a PhysicalConnection role class from which those implementations can inherit. An IE supporting the PhysicalConnection role must contain at least one Plug instance in order to be able to link it to Socket instances.

Logical Connection: A logical connection is always modelled between EIs of devices. The interfaces have to be of the type *LogicalEndPoint*. As for the physical connections, the library contains a LogicalConnection role which can be supported by logical connection IEs. Additionally, an IE supporting the LogicalConnection role has to contain at least one LogicalEndPoint interface in order to be able to link it to the interfaces of devices.

For the logical and physical endpoints, technology specific interface classes should be implemented. The described EIs used for physical or logical connections should be derived from those technology specific classes to enrich the model with more details. The original example of such classes is depicted in Figure 5.

Physical/Logical Network: The AML communication library contains the PhysicalNetwork and LogicalNetwork role classes to model the network topology. Such a topology is defined by its connections. Thus, an IE supporting the role PhysicalNetwork consists of IEs which support the role PhysicalConnection (analog for logical network and connections). Consequently, a physical or logical network is a container for the corresponding connections within the InstanceHierarchy.

A. Discussion

Due to the lack of rules and intuition in the application of the AML communication library it is unclear how to model network protocols and components. Even though the example of section III is neither complex nor dynamic (e.g. it does not consist of routing mechanisms and components or firewalls) several issues arise.

We found that the library is sufficient to model the physical layer of the ISO/OSI model. Nevertheless, the Physical$X$ role and interface classes were not adequate to model the data link layer (like defined in [5]). A simple example is the network separation using VLANs. Even though VLANs operate on the data link layer, they cannot be modelled with physical connections since they are realized on the logical level and use multiple physical connections.

This leads to another issue, the relationship between protocols is not considered in [5] and cannot be modelled with the current library. However, the information about such relationships is an essential fragment of modelling and building networks. Protocols make use of each other extensively and without the possibility to derive protocol stacks from an information model, this model is useless for other tools or will lead to false interpretation. Furthermore, to model security mechanisms, which was our original objective, a complete model of the underlying network including all protocol relationships is indispensable.

Consequently, clear rules of how to model protocols, their relationships and their impact on the network topology are needed. In section V we provide such rules and explain augmentations of the original library, which we implemented to ensure compliance to those rules.

V. Extending the AutomationML Communication Library

In section IV-A, we discussed the issues of modelling networks with the AML communication library. As a consequence, we provide a concept to solve those issues and present an extension of the original library which supports the implementation by additional role and interface libraries (cf. Figures 6, 7 and 8). Since the following paragraphs are easier to understand by having an example at hand, we create an AML model of the example network architecture from section III and explain selected segments of that model in section VI.

The concept is based on the established and well-known, layer-based ISO/OSI model (cf. I-A). By applying this model we support the way network architects, engineers and components separate technologies and protocols. However, our approach to utilize the ISO/OSI model slightly differs from the definition given in [5]. Instead of mapping the layers one and two to the Physical$X$ roles (see section IV), we only map layer one to these roles. We support this mapping by adding layer-based endpoint classes located in a library called IsoOsiLib and deriving technology specific endpoint interface classes (located in the libraries ProtocolsLib and HardwareInterfaceLib) from them respectively (see Figure 6). Based on this mapping, the following paragraphs introduce the extensions and describe the concept we followed, which can be used as collection of rules to add additional network protocols and devices.

Devices: Devices are modelled as explained in section IV. Additionally, we introduce roles of the class LogicalDevice located in a library called DevicesLib for each protocol or technology which differentiates a device from another just by supporting the protocol or technology. For example, as not all switches support VLANs we created a VlanDevice role (see DeviceLib in Figure 7). To model a switch which supports VLANs one can simply add VlanDevice as supported role to the switch's IE instance (cf. Figure 10).

Connections: As described in section IV, connections are modelled as IEs supporting either the PhysicalConnection or LogicalConnection role. To be able to connect protocol endpoints, we augment this concept by creating protocol specific connection roles (i.e. ConnectionsLib). Following our layer-based concept, the physical layer connections extend the PhysicalConnection role class and the other layers' connections the LogicalConnection role class. These new connection roles

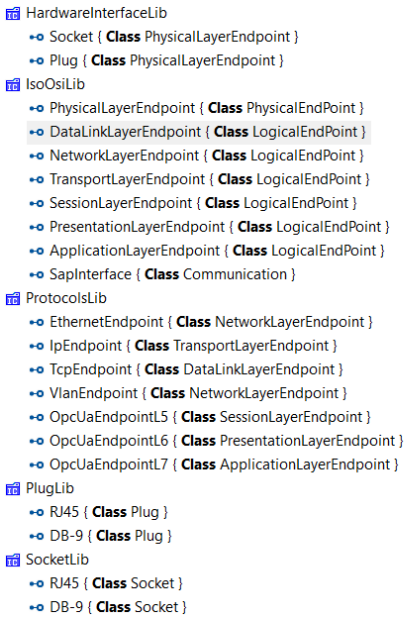

Fig. 6. Interface classes of the AutomationML communication library extension

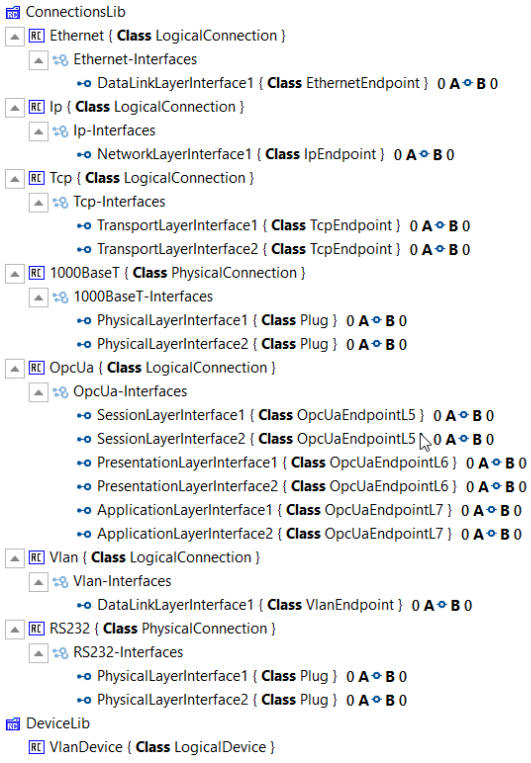

Fig. 7. Network Connection and Device role classes of the AutomationML communication library extension

contain interfaces which implement the respective endpoint classes. We demand that endpoints (like device sockets) are only linked to connection endpoints of the same endpoint interface class. The restriction to link only interfaces of the same type is already part of the basic concepts of AML [4]. Furthermore, we distinguish between single-layer and multi-layer connections which depends on the number of ISO/OSI layers they operate on and multiparty and end-to-end connections which describes whether an arbitrary number of endpoints can be connected or exactly two. In the following list the connection roles of Figure 7 are mapped to those types:

- Ethernet, IP and VLAN connections are single-layer multiparty connections. They have one interface which can be linked internally to each participating Ethernet, IP or VLAN endpoint.
- RS232, 1000BaseT and TCP connections are single-layer end-to-end connections. They have exactly two interfaces which are instances of the same endpoint interface class.
- An OPC UA connection is a multi-layer end-to-end connection. An OPC UA connection has two interfaces for each ISO/OSI layer OPC UA operates on (the suffixes L5, L6 and L7 refer to the respective ISO/OSI layer).

Networks: The paper [5] describes networks as a collection of connections. However, it does not define a rule to map connections to networks. Thus, we recommend to model networks close to reality by creating hierarchies. In such a hierarchy the root object is the overall network and the leafs are technology or protocol-specific connections. The children of the root object support protocol specific network roles and can consist of additional networks, supporting the same technology- or protocol-specific network role. How deep the respective encapsulation will be and what roles are used to isolate the hierarchy levels along the path from each other is technology or protocol dependent. This strategy is close to reality since, depending on the used protocol stack, one can, for example, create logically separated networks. These networks may again consist of sub-networks within which sub-groups of participating devices can exist that are isolated from each other.

Inter-Layer Relationships: With the concept depicted in the previous paragraphs, we are able to model networks on different protocol levels. Nevertheless, it is not yet possible to create protocol stacks and in particular protocol relationships. Our objective was to link a representative of one ISO/OSI layer to the representative of an adjacent ISO/OSI layer directly below or directly above. Such a representative has to be device-independent which is why linking endpoints of devices is not sufficient. However, we observed that connections meet the requirements to be such representatives. Furthermore, by linking connection endpoints directly, a parser or modeller would not be supported in ensuring that only adjacent layers are interconnected. Therefore, we introduce Service Access Point (SAP) role classes (cf. IsoOsiServiceAccessPointLib in Figure 8) and an interface class (cf. IsoOsiLib/SapInterface in Figure 6). When linking two connections of different layers, our concept requires to create an SAP IE which supports the respective SAP role from the IsoOsiServiceAccessPointLib and let it be a child of the higher layer connection's IE. Furthermore, it is required to add an external interface of the type SapInterface to each of the two connections. In addition, we demand that interfaces of the type SapInterface are only linked to SAP interfaces and vise versa.

Note: Due to the space limitations, we exclude attribute details like VLAN IDs or bandwidths in this paper.

Fig. 8. SAP role classes of the AutomationML communication library extension

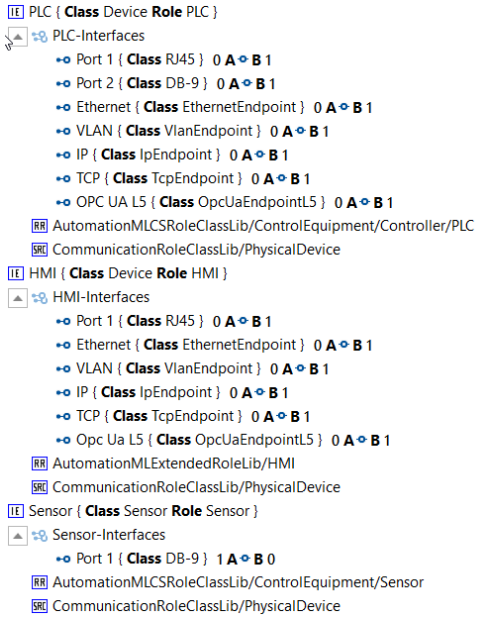

Fig. 9. InstanceHierarchy of devices PLC, HMI and Sensor with their respective endpoints and roles

VI. Example Model Using an Extended Communication Library

In this section we demonstrate the applicability of the proposed concept (cf. section V). For better comprehensibility we explain the different parts of section V on an AML model of the example network architecture from section III.

Devices: When modelling a device like the PLC (cf. Figure 9), we create an EI for each of its physical ports and assign the respective socket standard's interface class to it (cf. SocketLib Figure 6). In addition, we create an EI for every protocol and layer the PLC shall support. In our example the PLC uses Ethernet, IP, TCP and OPC UA on layer 5 (OPC UA on layer 6 and 7 was omitted since it is not relevant for our model). Furthermore, a VLAN EI was modelled to be able to add the PLC to a VLAN. It is essential that the protocol and technology-specific endpoints are derived from the IsoOsiLib endpoints. To create VLANs, Switch 1 and Switch 2 have to support the VLAN technology. This is indicated by means of supporting the role VlanDevice (cf. Figure 10).

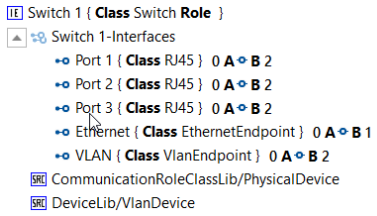

Fig. 10. InstanceHierarchy of Switch 1 with its endpoints end roles

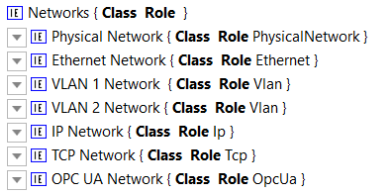

Fig. 11. InstanceHierarchy showing all network instances the model consists of

Networks: Figure 11 gives an overview of the networks we created for the example of section III. We modelled each VLAN as a separate network, whereas only one IP network IE exists. The IP network IE could also consist of several sub-networks, which is a classic example of the previously described hierarchy (cf. section V). Since our example is too small to contain sub-networks, the IP connection IE in Figure 12 is a direct child of the IP Network IE. The picture also shows that the roles NetworkLib/Ip and NetworkLib/Tcp are supported by the corresponding network IEs.

Connections: Figure 12 gives an example for two of the three connection types, i.e. two single-layer end-to-end TCP connections and one single-layer multiparty IP connection. All device IP endpoints in our example model are internally linked to the NetworkLayerInterface1 EI of IP Connection 1. The TCP Connection 1 has an interface TransportLayerInterface1 which is internally linked to the PLC's TCP endpoint and an interface TransportLayerInterface2 which is internally linked to the HMI's TCP endpoint. Analogously, the TCP Connection 2's transport layer interfaces are internally linked to the TCP endpoints of PC1 and PC2. The third connection type is depicted in Figure 13, which shows a multi-layer end-to-end OPC UA connection where we only linked the session layer interfaces to the respective OPC UA layer 5 endpoints of the HMI and the PLC (cf. Figure 9).

SAPs: In Figure 12 TCP Connection 1 has an interface towards the upper layer called SapInterfaceL5 and one towards the lower layer called SapInterfaceL3. To model TCP over IP (often referenced as TCP/IP), we created the Sap34 IE which supports the Layer3Layer4Sap role. This role has two predefined interfaces derived from the network layer

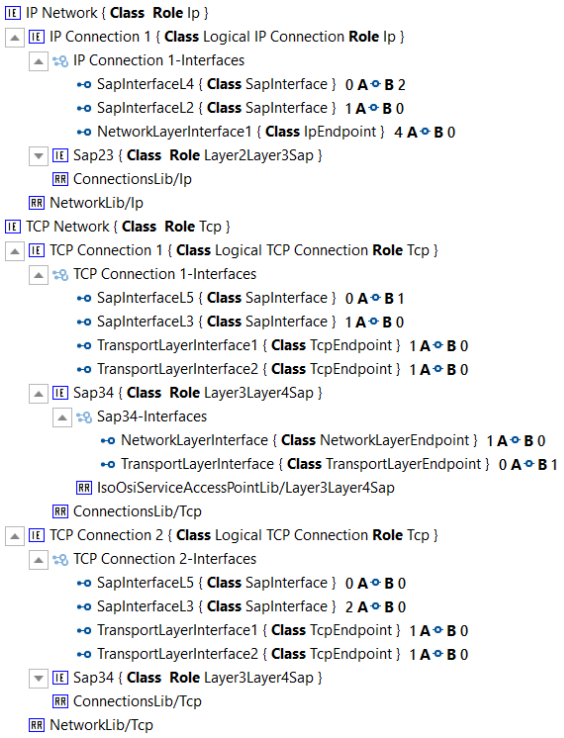

Fig. 12. InstanceHierarchy of the TCP and IP networks and connections with their roles, endpoints and SAP instances

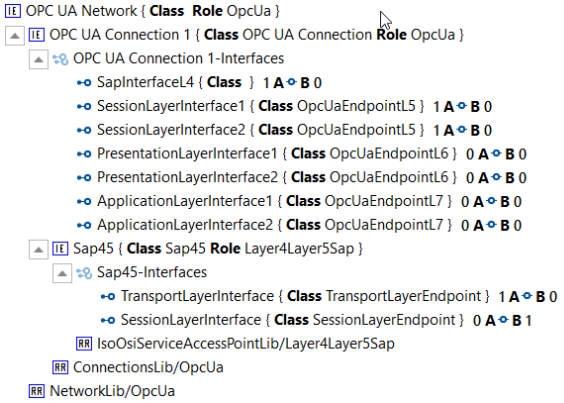

Fig. 13. InstanceHierarchy of the OPC UA network and its connection with the respective roles, endpoints and SAP intstances

and the transport layer interface classes. To build the actual protocol relationship, we created an internal link from the SapInterfaceL3 EI to the TransportLayerInterface of the Sap34 IE. Furthermore, we created an internal link from the NetworkLayerInterface EI of Sap34 to the SapInterfaceL4 EI of the IP Connection 1 IE. These steps fully establish a semantic relationship between the two layer representatives TCP Connection 1 and IP Connection 1.

We mainly modelled protocols and their inter-layer relationships. However, VLAN is an example for a technology or protocol feature. Thus, a VLAN connection has no own interface towards other ISO/OSI layers and relies on a protocol for this. To build the relationship between a VLAN connection and its protocol instance we equip it with an SAP EI towards its own

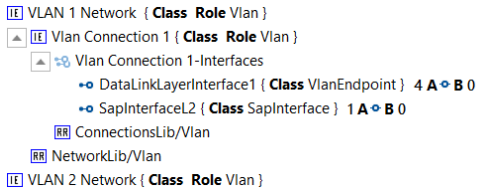

Fig. 14. InstanceHierarchy of the VLAN networks with the respective connection, role and interfaces of VLAN 1

layer (cf. SapInterfaceL2 in Figure 14). With this interface we are able to internally link it to Ethernet connections (via an SapInterfaceL2 EI of the Ethernet connection). Since the link is layer-intern, no SAP element is required.

VII. Related Work

The NETCONF Data Modelling Language Workgroup introduced the data modelling language YANG [17] to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF). Ongoing acitivities of the workgroup include a draft, which is dealing with network topology modelling and covers layers 1 to 3 of the ISO/OSI model [18]. While YANG is a completely generic modelling language, the draft provides additional types covering layers 1 to 3 of the ISO/OSI model. The key differences to our work are the scope and the level of detail. As our future aim is modelling security protocols and properties of networks, we need to cover all layers of the ISO/OSI model, as proposed in this paper. In terms of the level of detail, our library extension contains building blocks for various concrete protocols such as IP, TCP or OPC UA. Those building blocks enable the modelling of inter-protocol relationships, protocol support on specific devices and a concrete mapping of protocols and features to all ISO/OSI layers. Furthermore, it is possible to model layer-specific properties of a multi-layer protocol like OPC UA. However, recent approaches of Pfrang and Kippe [19] to support event correlation for intrusion detection show the relevance of modelling networks with YANG to apply smart security mechanisms. Given these related works, a mapping of our AML network modelling methodology onto YANG and respective tool support seems to be a reasonable future step.

Model-Driven Networking (MDN) [20] can be used to generate SDN-based (Software-Defined Networking) networks given a proper model. Domain-Specific Modelling Language (DSML), also proposed in [20], can be used to create such a model. Unfortunately, DSML cannot be linked to AML component models and is not sufficient to model necessary details like inter-layer relations. Nevertheless, MDN over DSML could be used as transformation from an AML model to an SDN source code.

The need for a methodology to model communication in AML is not new. In 2013 a paper was published addressing this issue [21]. Subsequently, it was refined as the whitepaper in 2014 and published by the AutomationML Consortium [5] (cf. section IV). Later on this methodology was refined again [22] and has already been used to create exemplary network components [23]. However, as we explained in section

IV-A their approach is not yet sufficient to model common communication networks and security properties.

VIII. Conclusion

We proposed an extension to the AML Communication whitepaper and its standard libraries, which adds the role and interface classes required for fine-grained modelling of network protocols and their interdependencies. It is based on the established ISO/OSI model whereby protocols of adjacent layers are interconnected via elements supporting service access point roles. If a device deploys a certain protocol or technology, we let it support the corresponding role class. Connections on the different layers are modelled via elements supporting certain connection roles. Connection roles implement endpoint interfaces which must only be connected to endpoints of the same interface class. Intuitively, networks are modelled hierarchically as trees. The root element is the overall network, and the leafs are protocol specific connections. Children of the root support either protocol- or technology-specific network roles and can consist of further networks supporting the same technology- or protocol-specifc role.

Our extension enables engineers to model networks on a level of detail, which is adequate for considering security by design and secure extension or maintenance of the networks. The resulting models allow in-depth security analysis, consistency checks of what has been planned against what is actually deployed in a production network and to derive configurations for switches, endpoints and, to a certain degree, routers and firewalls.

A key task in our future work will be a concept for modelling dynamic network components such as firewalls or routers, which requires the integration of semantic rules. We are also engaged in the AutomationML standardization activities so as to incorporate our work into future versions of the standard.

References

[1] R. Drath, A. Lüder, J. Peschke, and L. Hundt, "Automationml-the glue for seamless automation engineering," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. IEEE, 2008.

[2] T. Moser and S. Biffl, "Semantic tool interoperability for engineering manufacturing systems," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. IEEE, 2010, pp. 1–8.

[3] T. Moser, R. Mordinyi, D. Winkler, M. Melik-Merkumians, and S. Biffl, "Efficient automation systems engineering process support based on semantic integration of engineering knowledge," in *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*. IEEE, 2011, pp. 1–8.

[4] IEC, "IEC 62714-1:2014 Engineering data exchange format for use in industrial automation systems engineering - Automation markup language - Part 1: Architecture and general requirements," *IEC standard*, 2014.

[5] AutomationML Consortium, "AutomationML Whitepaper Communication," *AutomationML - The Glue for Seamless Automation Engineering*, 2014.

[6] ISO, "IEC 7498-1: 1994 information technology–open systems interconnection–basic reference model: The basic model," *ISO standard ISO/IEC*, pp. 7498–1, 1994.

[7] AutomationML Consortium, "Application Recommendations: Automation Project Configuration," *AutomationML - The Glue for Seamless Automation Engineering*, 2016.

[8] D. D. Nelson and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes," RFC 5080, Dec. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc5080.txt

[9] K. Seo and D. S. T. Kent, "Security Architecture for the Internet Protocol," RFC 4301, Dec. 2005. [Online]. Available: https://rfc-editor.org/rfc/rfc4301.txt

[10] "Transmission Control Protocol," RFC 793, Sep. 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc793.txt

[11] "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc791.txt

[12] T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5246.txt

[13] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Transport Layer Protocol," RFC 4253, Jan. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4253.txt

[14] E. Rescorla, "HTTP Over TLS," RFC 2818, May 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2818.txt

[15] AutomationML Consortium, "Whitepaper AutomationML Part 1 - Architecture and general requirements," *AutomationML - The Glue for Seamless Automation Engineering*, 2016.

[16] IEC, "IEC TR 62541-1:2016 OPC unified architecture - Part 1: Overview and concepts," *IEC standard*, p. 26, 2016.

[17] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020, Oct. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc6020.txt

[18] J. Medved, N. Bahadur, H. Ananthakrishnan, X. Liu, R. Varga, and A. Clemm, "A Data Model for Network Topologies," Internet Engineering Task Force, Internet-Draft draft-ietf-i2rs-yang-network-topo-12, Mar. 2017, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-i2rs-yang-network-topo-12

[19] J. Kippe and S. Pfrang, "Network and topology models to support ids event processing," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, 2017, pp. 372–379.

[20] F. A. Lopes, M. Santos, R. Fidalgo, and S. Fernandes, "Model-driven networking: A novel approach for sdn applications development," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 770–773.

[21] M. Dehof, A. Lüder, and M. Heinze, "An approach for modelling communication networks in industrial control systems," in *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE*. IEEE, 2013, pp. 7702–7707.

[22] F. Bendik and N. Schmidt, "Exchange of engineering data for communication systems based on automationml using an ethernet/ip example," in *ODVA Industry Conference and 17th Annual Meeting, Friso, Texas, USA*, 2015.

[23] A. Lüder, N. Schmidt, and M. John, "Lossless exchange of automation project configuration data," in *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, 2016, pp. 1–8.