J. Michael, J. Pfeiffer, A. Wortmann (Hrsg.): Modellierung 2022 Satellite Events,
Digital Library, Gesellschaft für Informatik e.V. 44

Development of a SysML Profile for Network Configurations in Safety-critical Systems

Andreas Hemmert[1] Andreas Schweiger[2]

Abstract: The contribution describes an approach for defining avionics network architectures in an existing system model. To this end, a SysML profile is developed containing stereotypes to specify such a network. The corresponding model captures the network's configuration. Such a configuration defines a network of a safety-critical complex system. As a result, systems engineers can use the profile in the systems development process with digital continuity. An example demonstrates the application of the profile in the model development.

Keywords: Avionics; Systems Engineering; Software Engineering; Model-based Development; Requirements; Qualification; Certification; Simulation; Process; Tool; Platform; Architecture; SysML

1 Introduction

1.1 Context

For the development of safety-critical complex systems highest quality standards are required. Applying these, improving productivity, and reducing development time are of uttermost importance in the limited avionics market with an increasing quantity of competitors. Therefore, creating digital twins of their products leads to the reduction of the time to market. Digital twins capture the digital representation of a real product and relating product information [Ku17]. These require models, that are exchangeable between different engineering disciplines. An essential key for digital twins is Model-based Systems Engineering (MBSE). MBSE means the central data management of all product-related information in models created during the systems development process [Ei14]. Consequently, a standardized modeling language is needed to support this approach and to create tool-independent models [Wa16].

OMG's SysML™ can be regarded as an enabler for MBSE, while taking into account the drivers mentioned above [Obb]. These language requirements are met by SysML: "The system model expressed in SysML provides a cross-disciplinary representation to enable integration with other engineering models and tools" [Obb]. Basically, SysML is an extension of a subset of elements and diagrams defined in the Unified Modeling Language (UML)

[1] Airbus Defence and Space GmbH, Manching, Germany, andreas.hemmert@airbus.com
[2] Airbus Defence and Space GmbH, Manching, Germany, andreas.schweiger@airbus.com

standard [Obj17b]. UML defines elements and diagrams used for software development. SysML targets the discipline of systems engineering, while reusing metaclasses of UML. The stereotypes, diagrams, and relations between metaclasses of the UML extension are defining the SysML metamodel [Obj19].

In the context of the presented systems development approach, a system model is created using SysML. The needs of the system are analyzed and necessary system components are designed. These components are communicating via networks. The network architecture is specified in parallel to the system model. Currently no data continuity exists between the system model and the network architecture model. Based on the previous statement of integrating models using SysML, the connection of the system model with the network architecture model seems to be achievable. Furthermore, the integration of these discipline models into a single one is an essential step towards developing a digital twin. In order to achieve the integration of the network architecture model into the system model, analyses have to be performed. They have to reveal, whether SysML (v1.6) is providing suitable means (i.e. SysML stereotypes) required for network configurations.

1.2 Research Question

As described in Sec. 1.1, the system model is developed for a safety-critical complex system. However, when it comes to network configurations, no data continuity to system models exists. In order to integrate the network architecture and the configuration of the communication into the system model, existing SysML capabilities are analyzed. As a result, the research question “What is the gap of elements and properties between SysML standard v1.6 and the required classes for network configuration?” will be answered [He20]. The taken approach and results are described in this contribution.

1.3 Related Work

Shames and Sarrel developed a pattern for defining network architectures [SS15]. The approach for defining communication ports for each network participant for each applicable Open Systems Interconnection (OSI) Layer is required for defining the network architecture. While Shames and Sarrel are providing means for defining network architectures, the ones for configuring a network are missing. Thus, our contribution finds suitable means for network configurations with SysML.

2 Selection of Network Architecture Classes

2.1 Method for Developing Network Architectures

Because SysML is offering many redundant system development means, a method has to be developed, which selects suitable stereotypes for developing the network architecture. Furthermore, such a method needs to define the model element organization of the system model. In addition, it has to specify the dedicated usage of SysML elements for the relevant steps of the development process. Two basic principles for network architecture development are found in literature and used in the approach of this contribution: **(1)** Separation into views: A network architecture can be defined by the requirements, functional, logical, and physical aspects (i.e. views or viewpoints) [St18, Bo21]. According to ISO 42010 [ISO11], a view is defined as a "work product expressing the architecture of a system from the perspective of specific system concerns" [ISO11, p. 2]. The view principle can also be seen as a pattern: Dividing a design into views offers the opportunity to focus on problem specific needs in a view with reduced model properties. The discussion in section 2.2 and the developed profile is organized according to the views Requirements, Functional, Logical, and Physical as suggested in the research project SPES 2020 [Po12]. **(2)** Abstraction of the communication into different layers: This principle is covered, e.g. in the OSI basic reference [Int94]. The OSI model is created based on the layer pattern for reducing the development complexity.

2.2 Selection of Classes to Express Network Architectures

Classes have been introduced in the object-oriented software engineering paradigm. A class in this domain is a template of methods and attributes for an object category [Jo17]. Nevertheless, the principle is useful for defining systems, as well. A system usually consists of functionality (i.e. class methods) and attributes denoting and storing the state of the system. This Sec. introduces the selected classes that are required to describe network architectures. The selection of classes is necessary, in order to avoid ambiguities of the same aspect. This step limits also the quantity of different stereotypes. The selection is done by analyzing each class for the context of network configuration of a safety-critical complex system. The classes that are discussed in this Sec. are indicated by using the format `Classname`.

The Functional View is consisting of `Tasks` [Eb15, p. 605] connected via `Dataflows`. A `Task` is selected instead of a `Function`, because `Function` is more suitable at system abstraction level and independent of its implementation [SAE10, p. 11]. `Dataflows`, which are the only identified means, are defining the functional interfaces [Eb15, p. 605]. The functional interface description is important in the context of a safety-critical system: The dataflow analysis is performed in a safety assessment and analyzes the functional

dependencies. Dependent on the safety-criticality of a function one needs to ensure, that a failure of the function must not lead to failures of other functions.

Building the bridge to the Logical View, `Tasks` are allocated to an `<OSI Layer> Entity` [Int94]. The angle brackets are placeholders for the respective OSI Layer. The entity is named dependent on the contextual OSI layer. `<OSI Layer> Entity` is generic, whereas `Module` is related to modular architecture [FE17, p. 1942]. Therefore, `<OSI Layer> Entity` is selected. The entities are connected via `Virtual Links` to express the logical connection [Ri17, p. 5]. `Virtual Link` is selected, because of the highest occurrence in literature. Furthermore, because of the term virtual it becomes clear, that the link is not a real connection. Instead, it represents one or multiple connections. `Auxiliary Link` [Yu19, p. 2655] could be used instead, but is not found that often in literature.

A `Node` is a generic name for a network element of the physical view [STVH19, p. 6268]. `Node` can be used instead of `Module`. Modules shall be used, if the system is modular and flexible. Flexibility in a critical system is important, whereas all configurations need to be certified. `Nodes` can be further specialized, e.g. for denoting the functionality of a `Bus Controller`[3] [Pl18, p. 114]. The specializations can vary depending on the project context.

A `Network` is a class required for grouping connections and is implemented with `Switches` [Yu19, p. 2654] or `Buses` [Ch16, p. 206] together with their necessary `Communication Protocols`. While a `Bus` connects `Nodes` located all over the system, a `Switch` is in a `Zone` [Eb15, p. 605], which denotes a location in the system. A `Node` can be part of a `Zone` or `Domain` [St18, p. 73]. In order to connect the `Node` to the `Network`, a `Network Port` is necessary. The selection of `Switch`, `Bus`, `Domain`, `Zone`, and `Network Port` is straightforward.

3 Transition from Classes to SysML Stereotypes

The classes selected in Sec. 2.2 need to be mapped to SysML stereotypes. This is performed based on the class nature (e.g. structural or behavioral). As a preparation, SysML diagram and element types are allocated to the views of the view pattern according to Pohl et al. [Po12], which simplifies the mapping.

3.1 SysML Stereotypes for Functional View Classes

System functions relevant for the Functional View are expressed via `SysML::Activities` [Po12]. Activities define the transformation behavior from input to output parameters [FMS14]. A `SysML::Activity Diagram` is used to define the functional

[3] "The Terminal assigned the task of initiating information transfers on the data bus" [U.S18, p. 2].

interactions and interfaces via SysML::Object Flows [Obj19]. SysML::Control Flows could be used, as well. However, SysML::Object Flows define the data exchanged between functions, while the data (object) interfaces between functions are again essential for a safety assessment. Functions could be expressed as SysML::Blocks instead [Obj19]. SysML::Blocks are powerful, when it comes to specializations, generalizations, and system breakdown. However, the focus in the Functional View lies on functional interfaces and dependencies. Therefore, no blocks are used in this view. For allocating the classes of Sec. 2.2 to the SysML elements of this view, Tasks are implemented with SysML::Activities. The functional interfaces that are implemented with Dataflow classes can be expressed by SysML::Object Flows. A SysML::Object Flow is related to SysML::Activity Parameters. Whereas SysML::Parameters defines the interface of a SysML::Activity, the SysML::Object Flow defines the connection to another SysML::Activity. SysML::Parameters need to be typed by SysML::Value Types to express the kind of data exchanged via the SysML::Parameter [Obj19]. The resulting stereotype extensions are shown in Fig. 1

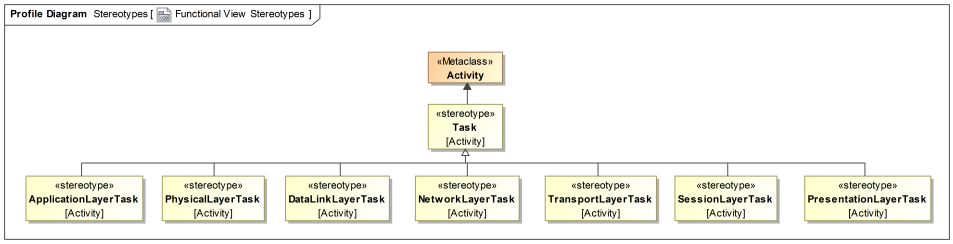

Fig. 1: Functional View Stereotypes

3.2 SysML Stereotypes for Logical View Classes

SysML::Blocks denote logical and physical elements of each abstraction level [Po12]. Hence, the physical and logical system decomposition is implemented using SysML::Blocks [Obj19]. The SysML::Block decomposition is expressed by UML::Composite Aggregations. The SysML::Block Definition Diagram is the most suitable diagram type to authorize and decompose blocks. Alternatively, tables can be used to define the part associations [Obj19] in a semantically equivalent way. A SysML::Block expresses the Logical View element <OSI Layer> entity. The SysML::Block Definition Diagram is used to relate the <OSI Layer> entity as parts of the logical parts, i.e. items by using UML::Composite Aggregations. Whereas specializations of SysML::Block are used for elements introduced by the network architecture model, specializations of UML::Named Element can be used for elements that are already defined in the system model. This is, because stereotype specializations of SysML::Block cannot be applied to existing block elements.

The SysML::Internal Block Diagram is introduced with SysML to define the internal relations of system elements [Obj19]. Particularly the interface definition of a system and connections to other systems is defined with the SysML::Internal Block Diagram. While the Logical View shows only logical connections, the Physical View visualizes real connections in a system in this diagram type. Logical interfaces are expressed via SysML::Proxy Ports [Obj19]. This stereotype is adequate, because it adds no additional part properties to the system definition. Instead, a SysML::Proxy Port represents the aggregated real interfaces via several hardware elements. In order to specify the data exchanged via the ports, a SysML::Flow Property and SysML::Item Flow can be used. SysML::Flow Properties are properties of a SysML::Block specifying data items flowing into or out of SysML::Block or both [Obj19]. Each SysML::Flow Property represents a data item that can flow into or out of a block. A SysML::Flow Property could be a property for each SysML::Block denoting a system element.

In the example of a computer, a SysML::Flow Property could be a video signal provided to a display. Connecting two SysML::Blocks, the respective SysML::Flow Properties need to match. This means, the display block needs to have a video signal flow property, as well. Other system elements connected to the computer are not requiring the video signal. Therefore, it is more suitable to attach the flow property to a port. This is implemented by defining a SysML::Interface Block.

Similar to the SysML::Block, the SysML::Interface Block can have SysML::Flow Properties. Because the SysML::Interface Block is specifying the flow behavior of the port instead of the whole block, it limits the potential receivers of the data items. Again, in the computer example one port is expressing the display interface. It is typed by an interface block containing the video signal flow property. Another computer port is typed by an interface block containing the power signal flow property.

SysML::Item Flows are like flow properties. SysML::Item Flows can be attached to SysML::Connectors [Obj19]. Due to this fact, a port can have multiple connections with different items flowing on each connection. In the computer example this could be a voltage meter attached to the display interface using the same port. Whereas the interface block contains a video signal, i.e. voltage and current flow properties, the voltage meter only receives the voltage information[4]. SysML::Item Flows can be linked to SysML::Activities [Obj19]. This is a useful feature in the view pattern, because the Functional View (SysML::Activities) can be linked to the Logical and Physical View (SysML::Blocks exchanging SysML::Item Flows). The element contained in the SysML::Item Flow is called conveyed item and can be a SysML::Block. A SysML::Block can have SysML::Signals. SysML::Signals can be used to

[4] In fact, a voltage meter receives a small amount of current, as well. Due to the high impedance of the voltage meter, this current can be assumed to be 0 A.

define Interface Control Data (ICD) messages. Attributes of a SysML::Signal can express the actual ICD signals as part of an ICD message [FMS14, p. 146].

A Virtual Link expresses the routed connection between two end points independent from the physical connections. Therefore, it should be expressed as a SysML::Connector. However, a SysML::Connector simply describes, that there is a connection, but it is not specifying, which objects are exchanged. In order to specify the content flowing via the connections, SysML::Item Flows are used. The Virtual Link can be expressed as a conveyed item of a SysML::Item Flow. Alternatively, SysML::Interface Blocks can be used to type the SysML::Proxy Ports and contain SysML::Flow Properties. The SysML::Flow Properties contain information that can be provided via the typed SysML::Proxy Port. Both the SysML::Interface Block and the SysML::Block can be attached as conveyed items to the SysML::Item Flow, in order to specify the connection. However, increased effort is expected by applying both. Therefore, the SysML::Item Flow is used only, because it describes the actually flowing items. The messages exchanged via the Virtual Link are contained in a SysML::Block as SysML::Signals. Each signal has attributes representing the variables exchanged via the SysML::Item Flow. The attributes are typed with SysML::Value Types. The resulting stereotype extensions are shown in Fig. 2. It illustrates the extension of UML::Named Element instead of SysML::Block for the model elements as explained above.

3.3 SysML Stereotypes for Physical View Classes

Real interfaces, which are necessary for the Physical View (see Fig. 3) are expressed via SysML::Full Ports [Obj19]. In contrast to the SysML::Proxy Ports full ports are expressing real physical parts of the system. E.g., the SysML::Proxy Port expresses, that there is any interface type from a computer to the keyboard. The SysML::Full Port specifies, that the interface is implemented via a Universal Serial Bus (USB) Port.

By mapping the Physical View classes to stereotypes, it can be seen, that SysML::Blocks are beneficial, if it comes to decomposition and interface definition. Therefore, a Node and a Module is expressed using a SysML::Block. Nodes are set in relation to Domains and Zones using UML::Composite Aggregations. Thus, Network, Domain, and Zone are SysML::Blocks. A Node can be a part of a Network and a Zone at the same time. The same applies to the Domain and its parts.

Network Ports are expressed using SysML::Full Ports, because they exist as real parts of a Node. Network Ports are connected using SysML::Connectors authorized in a SysML::Internal Block Diagram. SysML::Interface Blocks representing different connection types are used to type the Network Ports. Possible connection types are Ethernet, Electrical, Fiber Channel etc. Network Ports are sometimes not peer-to-peer connections, which is dependent on the network type.

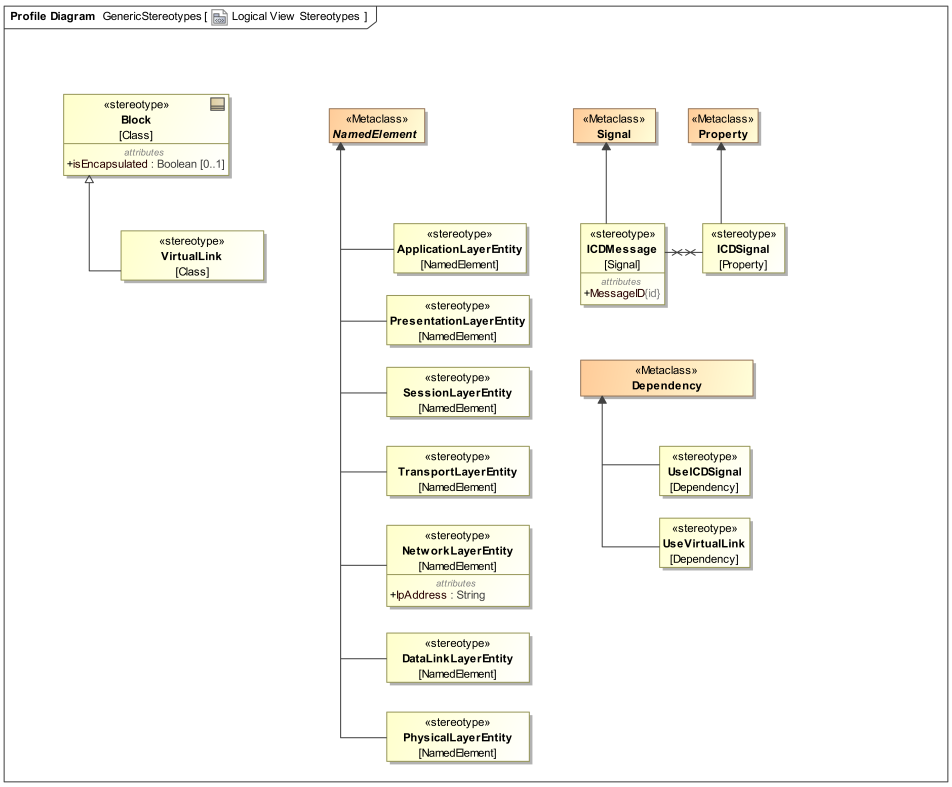

Fig. 2: Logical View Stereotypes

Therefore, `Switch` and `Bus` are introduced via `SysML::Blocks`, as well. For full detailed modelling, a communication protocol behavior can be added to the `SysML::Interface Block` by `SysML::State Machine Diagrams`. `SysML::State` machines are used to describe the different states of a network node in a network [SS15]. However, this approach is not analyzed yet as part of this contribution.

4 Application and Verification of Method and Stereotypes in a Demonstrator Model

A demonstrator model[5] is developed, in order to apply and to verify the method and stereotypes. Cameo Systems Modeler 19.0 is used to define the system and the network architecture. As the demonstrator requests further variables to be defined, further specializations of the

[5] For the verification purposes a demonstrator model is necessary, because it evaluates the applicability of the developed profile. This model can also be regarded as an example.

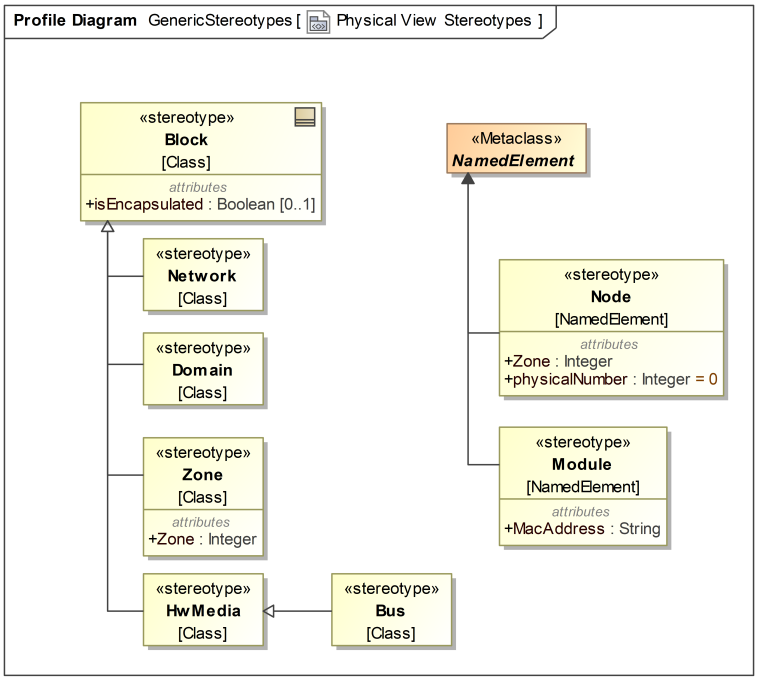

Fig. 3: Physical View Stereotypes

stereotypes are defined. They are «`NetworkCard`» (extending «`NamedElement`») and «`NetworkNode`» (specializing «`Node`»). The example model defines a network, which is consisting of network nodes. The network nodes consist of network cards to access the network. Information of several OSI layers are defined at the network cards. Fig. 4 shows the relations between the model elements of the Physical View. An excerpt of the connections between the network nodes are illustrated in the internal block diagram in Fig. 5. The diagram shows a connection implementing the MLD-STD-1553B bus communication and an adapter for adapting the Positioning Unit. The adapter is necessary, because just an Ethernet interface exists.

In the logical view, end-to-end communication is defined. This relates to OSI layer 3 (Network Layer). Therefore, Network Layer Entities are defined. The end-to-end-communication is defined by ports and connectors. Item flows are realized by the connectors to attach the traffic to the connectors. The traffic is defined by the `Virtual Link Block` with relations to ICD messages. Fig. 6 illustrates the example of the Logical View, where «`Network Layer Application`» is a specialization of «`Network Layer Entity`».

As a result, the network configuration approach using the developed profile has been verified by the demonstrator model. However, the final application of the approach and thus its full

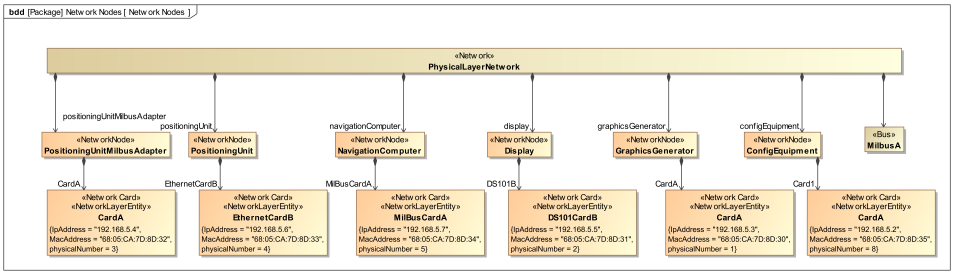

Fig. 4: Network Nodes

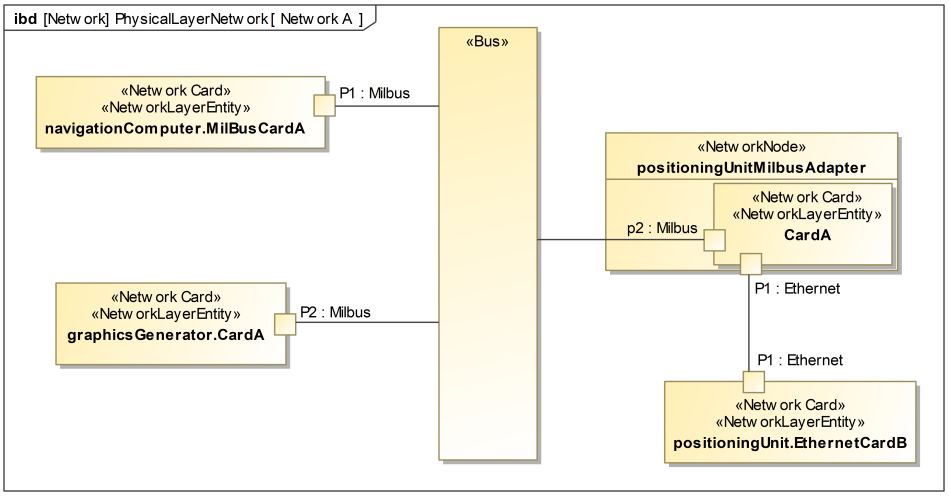

Fig. 5: Example Physical Layer Network Connections

verification including functional view is left to be performed, once the complete system model of the real product is available.

5 Fit-gap Analysis between Tool, Metamodel, and SysML v1.6

As the result of the analysis to select a suitable tool, Cameo Systems Modeler based on SysML v1.5 [Obj17a] is chosen. This Sec. analyzes the differences between v1.5 and v1.6 for SysML stereotypes applicable for the stereotypes of the developed profile. Furthermore, differences between the metamodel of Cameo Systems Modeler and SysML v1.5 are analyzed. The result is the traceability of differences between Cameo Systems Modeler to the current SysML standard v1.6. The reworked changes between SysML v1.5 and v1.6 are published by the authorizing standards body OMG. In this Sec. each issue is referenced by the key numbers of the changes given in [Oba] (e.g. SYSML 16-001).

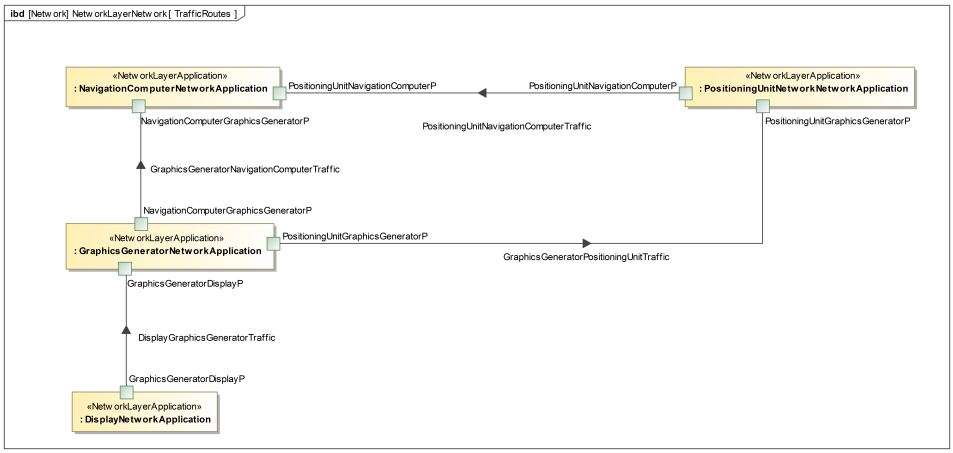

Fig. 6: Example Logical Layer Network Connections

In SysML v1.6 the constraint of locating a `SysML::Instance Specification` in the same `SysML::Package`, where it belongs to, is deleted (SYSML16-185). `SysML::Instance Specifications` are required for the usage of expressions, e.g. `UML::Opaque Expressions`. `UML::Opaque Expressions` are required to express hexadecimal values. In Cameo the constraint of SysML v1.5 is strictly implemented, so that a `SysML::Instance Specification` is only allowed to be owned by a `SysML::Package`. Subsequently, a `SysML::Instance Specification` cannot be nested in a `SysML::Block` it belongs to. This means, the value itself is decoupled from the usage. This may lead to a confusion of the engineer during the configuration of the system. The configured value shall be located under the `SysML::Block` to see the dependency already by the “own”-relationship. If the Cameo metamodel is switched to SysML 1.6, it should be possible to add a `SysML::Instance Specification` to a `SysML::Block`. Thus, the configuration of networks using `SysML::Value Properties` instead of stereotype tags simplifies the definition of hexadecimal values with `UML::Opaque Expressions`.

Since SysML v1.6 it is not required anymore to specialize each part association of a `SysML::Block`, if the block itself is specialized by another block (SYSML16-154). This impacts the network profile in the specialization of `Physical Parts` by `Nodes`. The workload of the tool user is dependent on the number of mouse clicks. Therefore, it saves time and avoids consistency conflicts, if the metamodel is adapted to SysML v1.6.

No further differences between the Cameo metamodel and SysML v1.5 are identified. The analysis needs to be repeated for future versions to ensure SysML standard conformance.

6 Answer of Research Question

The identified gaps between SysML v1.6 and required classes for network configurations are organized in the Functional, Logical, and Physical Views. In the Functional View, the `Task` is a specialization of the `SysML::Activity`. The main features of a `SysML::Activity` are suitable for defining the behavior of the logical elements (e.g. Data Link Entity) by connecting them with `SysML::Object Flows`. However, the Functional View is not reflected in the verification through the demonstrator model (see Sec. 4). The demonstrator model requires the configuration of the Logical and Physical View. The Functional View is ignored. Therefore, it could differ from the actually implemented behavior of the logical elements. Finally, the behavior located in the Functional View has to be verified by further quality assurance methods (e.g. tests). A possible future scenario would be to host software code in the Functional View and create software builds from the model. This enables the execution of the model or its parts.

The Logical View also addresses each OSI layer like the Functional View. The `<OSI Layer> Entity` is used in each OSI layer. The `SysML::Block` is used, because of its structural features using part associations. It is observed, that the specialization of the item is not the best way to add additional information of the same thing. By specializing a `SysML::Block` all parts are inherited. However, a new block has to be created as the specialized block. The specialized block inherits all ports of the base `SysML::Block`. During the network configuration, ports have to be redefined. A redefinition of a port decouples it from the original port at the base block. In fact, a new block with ports is created. The behavior is not supporting the idea of having different views and layers, because the information is displayed redundantly. The SysML standard needs to be enhanced to support such a requirement. Another central stereotype in the Logical View is the `SysML::Item Flow`. The `Item Flow` is used in this profile to attach the `Virtual Link` (`SysML::Block`) as a conveyed item. Whereas the `SysML::Item Flow` specifies the items that flow from block to block, the `SysML::Interface Block` typing a `SysML::Proxy Port` or `SysML::Full Port` specifies the items, that can potentially flow. Using the `SysML::Interface Block` instead would lead to multiple ports, because each port has to be used for each connection. The idea of defining both leads to an increased effort depending on the project. Regarding the context of a safety-critical complex system, the effort could lead to delays in system development due to formally redundant specifications. There should be a trade-off for defining one instead of both. In this contribution the approach of specifying only a `SysML::Item Flow` by the conveyed item is used in the Logical View.

In the Physical View the same issues apply to block specialization as discussed in the Logical View. In here, `Nodes` are specializing `Physical Parts`. Additional `Network Ports` are added to the `Node`, which redefine the ports of the `Physical Part`. As a result, the `Network Ports` are decoupled from the ports of the Physical Part. Consequently, a change of the port at the Physical Part does not directly trigger a change of the `Network`

`Port`. In the context of safety-critical complex system, this leads to a gap of traceability, which is unacceptable.

Independent from the views SysML lacks a way to define number formats different to the decimal format. `SysML::Opaque Expressions` can be used for expressing the value within a dedicated language (e.g. JavaScript). SysML v1.6 is on the right track by deleting the constraint of locating the `SysML::Instance Specification` in the `SysML::Package`, where the block is owned.

Because the contribution is limited to the Network, Data Link, and Physical Layer, further analysis is required for upper OSI layers. These layers could impact the network configuration. Therefore, constraints between layers could need to be taken into account. The analysis is not part of this contribution.

As a result, all network architecture classes required for network configuration can be implemented with SysML. Verification of the selected classes implemented in stereotypes was successful. However, further verification needs to be performed by proof of concept based on different verification methods and network configuration tool vendors.

7 Consequences

7.1 Consequences for System Model Definition in SysML

The network profile developed in the contribution can be used for the specification of network architectures in SysML. The single application of the profile does not ensure a consistent network architecture definition. Therefore, a method is developed. The method of separating the model into different views ensures the addressing of stakeholders' specific needs. Furthermore, the model separation into OSI layers enables the management of the complexity of the network architecture development. Consequently, by the application of the method and the profile, the network architecture is defined.

Regarding the identified gaps discussed in the research question (see Sec. 6), SysML has to evolve to fulfill specific needs. One of the major drawbacks is the lack of possibilities to define values in different number formats. Even if SysML v1.6 introduces means to improve this issue, the issue itself is not solved yet.

Although SysML provides features to create views (e.g. by using the specialization relationship), the application is not trivial. This leads to blocking points and misunderstandings by engineers. SysML should improve the communication between engineers and avoid misunderstandings. The standard should evolve to tackle this issue to be more comprehensible. Alternatively, the focus has to be on educating the engineers.

For systems development decisions have to be taken, if all information shall be covered in a single model. Different modelling languages are evolving and address domain specific

needs. E.g., the authorization of ICD messages in the SysML tool has turned out to be not straightforward. Dedicated ICD management tools need to be used. The ICD messages could be imported to the SysML tool.

The developed profile is generic. However, it can only be verified, whether the model information can be reused. Therefore, the model is exported for further use (e.g. network configuration software). In the scope of the presented work a network configuration tool is developed by a tool vendor. Consequently, the verification is dependent on the information exchanged with the network configuration tool. The profile is adapted using stereotypes to enhance the generic stereotypes, in order to capture the information specific to the tool vendor.

7.2 Consequences for Model-based Systems Engineering

At the beginning of each system development process it needs to be decided, what the scope of the model is. In theory, all aspects of a system design shall be entered into the system model. Practically, this theory is limited by project constraints, e.g. costs or development time. However, it is expected that an increased effort in system modeling in the development phase (frontloading [Fa09]) will reduce costs in the production and service phases.

The network configuration based on the network architecture in the system model builds the bridge between the digital and the real dimension of the system. In the context of a safety-critical complex system, the required traceability from the specification of the system model to the network configuration is ensured now. Finally, the objective of creating a digital twin is getting closer.

8 Conclusion and Outlook

The presented approach is developed for SysML v1.6. Further adaptions could be required due to the release of SysML v2.

The scope of the verification is limited to Network, Data Link, and Physical Layer. However, the developed method intends to define upper OSI Layers, as well. It needs to be verified, whether these layers can be defined using the profile and method. Regarding upper OSI Layers the design is driven by the decision of message-based or content-centric communication design. For example, the Data Distribution Service (DDS) Standard is shifting the message-based communication to data-centric communication [DD]. The impacts on the selected classes (see Sec. 2.2) have to be analyzed to fulfill the needs of such technologies.

The network architecture is dependent on the given DDS Quality of Service (QoS) information defined in upper OSI Layers. Lower OSI Layers have to implement the achievement of a specific QoS. Therefore, further analysis has to be performed on the impact between the OSI Layers for such a technology.

Once the network architecture is defined mostly, further analysis is performed: As an example, the allocation of traffic routes to TDMA (Time Division Multiple Access) scheduled time slots is analyzed regarding the traffic load. The schedule is necessary, as it defines the network communication[6]. Therefore, a permutation of all possible configurations is generated and analysed. The approach is visualized in Fig. 7 and described with the following steps: **(1)** Logical and physical views of the network architecture are imported into a Permutation

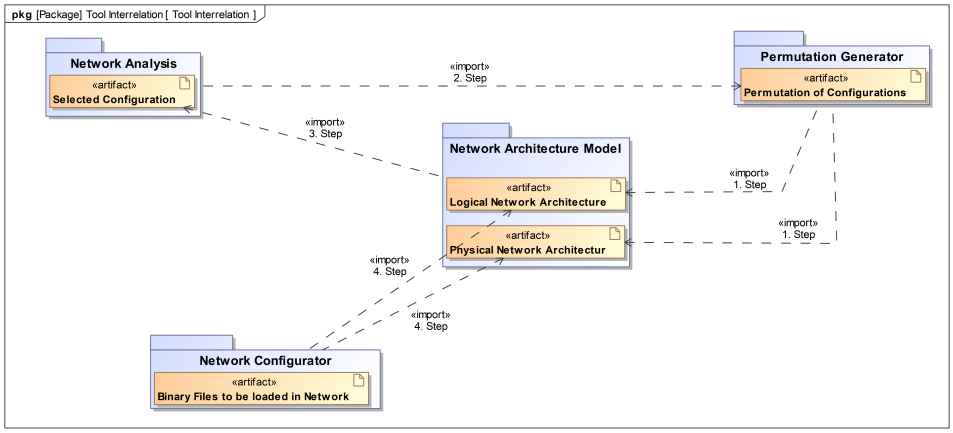

Fig. 7: Example Logical Layer Network Connections

Generator. The generator creates several possible configurations of virtual links to time slot allocations. A Traffic Route defines the End-to-End communication without mentioning all network nodes in the routing. **(2)** The generated configurations are imported into the Network Analysis Tool. The best configuration is proposed based on given criteria of the network calculus [WT]. **(3)** The selected configuration is imported into the Network Architecture Model. **(4)** Logical and Physical Architecture are imported into the Network Configurator to generate binary files, which are loaded into the network nodes: The network architecture consisting of the information above is exported into an XML file. The file is imported into the Network Configurator. It generates binary files, which are loaded into each network node of the real avionics application. This enables the respective nodes to communicate according to the specified patterns.

Most recent standardization documents require the coverage of cyber-security for aircraft certification purposes. Since this aspect is not yet covered by this contribution, relevant aspects along e.g. UMLsec [Ju10] have to be transferred to our approach as part of future work.

[6] The defined network communication is finally loaded into the network nodes of the real avionics application.

Bibliography

[Bo21] Boehm, W.; Broy, M.; Klein, C.; Pohl, K.; Rumpe, B.; Schroeck, S.: Model-Based Engineering of Collaborative Embedded Systems. Springer Cham, 2021.

[Ch16] Champeaux, P.B.; Faura, D.; Gatti, M.; Terroy, W.: A Distributed Avionics Communication Network. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W). pp. 206–209, 2016.

[DD] DDS Foundation: How does DDS work? https://www.dds-foundation.org/how-dds-works/, last access: 4th May 2022.

[Eb15] Ebeid, E.; Medina, J.; Quaglia, D.; Fummi, F.: Extensions to the UML profile for MARTE for distributed embedded systems. In: 2015 Forum on Specification and Design Languages (FDL). pp. 1–8, 2015.

[Ei14] Eigner, M.; Apostolov, H.; Dickopf, T.; Schäfer, P.; Faißt, K.-G.: System Lifecycle Management: Am Beispiel einer nachhaltigen Produktentwicklung nach Methoden des Model-Based Systems Engineering. Zeitschr. f. wirtsch. Fabrikbetrieb, vol. 109, no. 11:853–860, 2014.

[Fa09] Faust, P.: Lean durch Frontloading. Zeitschrift für wirtschaftlichen Fabrikbetrieb, 104(5):366–370, 2009.

[FE17] Feng, H.E.; Ershuai, L.I.: Deterministic bound for avionics switched networks according to networking features using network calculus. Chinese Journal of Aeronautics, 30(6):1941–1957, 2017.

[FMS14] Friedenthal, S.; Moore, A.; Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. The MK/OMG Press. Elsevier Science, 2014.

[He20] Hemmert, A.: Development of a SysML profile for network configuration in safety critical complex systems. Master's thesis, FOM Hochschule für Ökonomie and Management – Hochschulzentrum München, 2020.

[Int94] International Telecommunication Union. ITU-T Recommendation X.200: Data Networks and Open System Communications – Open Systems Interconnection – Model and Notation, 1994.

[ISO11] ISO/IEC/IEEE. 42010:2011 Systems and software engineering – Architecture description, 2011.

[Jo17] Jones, C.: Software Methodologies: A Quantitative Guide. CRC Press, 2017.

[Ju10] Juerjens, J.: Secure Systems Development with UML. Springer, 2010.

[Ku17] Kuhn, T.: Digitaler Zwilling. Informatik Spektrum, 40, no.5:440–445, 2017. DOI: https://doi.org/10.1007/s00287-017-1061-2.

[Oba] Object Management Group: OMG System Modeling Language - Closed Issues. https://issues.omg.org/issues/spec/SysML/1.6/fixed, last access: 4th May 2022.

[Obb] Object Management Group: SysML SPECIFICATION AT OMG – Introduction. https://www.omg.org/intro/SysML, last access: 4th May 2022.

[Obj17a] Object Management Group. OMG Systems Modeling Language (OMG SysML™), v1.5 edition, 2017.

[Obj17b] Object Management Group. OMG Unified Modeling Language (OMG UML), v2.5.1 edition, 2017.

[Obj19] Object Management Group. OMG Systems Modeling Language (OMG SysML™), v1.6 edition, 2019.

[Pl18] Plankl, H.: 6.1 – MIL-STD-1553B and it's potential for the future. In: ettc2018 - European Test and Telemetry Conference 2018-06-26 - 2018-06-28 Nürnberg, Germany. pp. 114–122, 01 2018. DOI: 10.5162/ettc2018/6.1.

[Po12] Pohl, K.; Hönninger, H.; Achatz, R.; Broy, M.: Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology. Springer Berlin, Heidelberg, 2012.

[Ri17] Ricker, T.: Avionics bus technology: Which bus should i get on? In: 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC). pp. 1–12, 2017.

[SAE10] SAE International. ARP4754 (R) Guidelines for Development of Civil Aircraft and Systems, 2010.

[SS15] Shames, P.M.; Sarrel, M.A.: A modeling pattern for layered system interfaces. 25th Annual INCOSE International Symposium (IS2015), Seattle, WA, July 2015.

[St18] Steinbach, T.: Ethernet-basierte Fahrzeugnetzwerkarchitekturen für zukünftige Echtzeitsysteme im Automobil. Springer Vieweg, 2018.

[STVH19] Sollfrank, M.; Trunzer, E.; Vogel-Heuser, B.: Graphical Modeling of Communication Architectures in Network Control Systems with Traceability to Requirements. In: IECON 2019 – 45th Annual Conference of the IEEE Industrial Electronics Society. volume 1, pp. 6267–6273, 2019.

[U.S18] U.S. Department of Defense (DoD). Digital Time Division Command/Response Multiplex Data Bus, 2018.

[Wa16] Wasson, C.S.: System Engineering - Analysis, Design, and Development. Wiley, Hoboken, New Jersey, USA, 2016.

[WT] Wandeler, E.; Thiele, L.: Modular Performance Analysis with Real-Time Calculus. `https://www.mpa.ethz.ch/`, last access: 4th May 2022.

[Yu19] Yunus, N.A. Md; Othman, M.; Hanapi, Z.M.; Yeah Lun, K: Enhancement Replicated Network: A Reliable Multistage Interconnection Network Topology. IEEE Systems Journal, 13(3):2653–2663, 2019.