Eclipse 4diac™ White Paper:

Modelling Network Communication in IEC 61499

29.08.2025

Friederike Bruns and Alois Zoitl
friederike.bruns@uol.de, alois.zoitl@jku.at

Contents

Figures

Tables

Foreword

Eclipse 4diac™ has been at the forefront of enabling distributed industrial automation based on the IEC 61499 standard. As an open-source project with a diverse and growing user and contributor base, we are committed to not only implementing the standard but also advancing its practical application in real-world systems.

While IEC 61499 defines a solid foundation for event-driven, component-based automation, its implementation leaves room for interpretation in several areas. In our journey developing Eclipse 4diac and supporting its adoption across industry and academia, we have frequently encountered challenges and design decisions not explicitly addressed by the standard. These include limitations in the language elements defined in IEC 61499, architectural trade-offs, integration strategies with existing systems, performance optimizations, and opportunities to extend the standard's capabilities.

The Eclipse 4diac white paper series aim at closing this gap. They serve two primary purposes:

a) Documenting key implementation decisions made in Eclipse 4diac that go beyond the standard's scope.

b) Providing a platform to discuss and propose potential extensions or refinements to the IEC 61499 specification, grounded in practical experience.

As with the project itself, these white papers are intended to be open and collaborative. We welcome feedback, critique, and contributions from the wider community. Whether you're a developer, user, researcher, or standardization expert, your insights can help improve Eclipse 4diac and shape the ongoing development of IEC 61499.

We invite you to engage with these documents as a resource and as a starting point for conversation. Together, we can ensure that the tools and standards driving next-generation industrial automation are both robust and adaptable.

Eclipse 4diac White Paper: Modelling Network Communication in IEC 61499 — Friederike Bruns and Alois Zoitl — August 2025

1 Scope

Developing industrial distributed automation systems demands high efforts from engineers due to increasing complexity. This particularly results from the required communication between the involved devices. To handle this complexity, industrial distributed control software is often designed using modeling languages such as the one defined in the IEC 61499 standard or in the Systems Modeling Language (SysML). IEC 61499 aims to ensure a platform-independent application model that can adapt to new hardware configurations. However, re-configuring the hardware also requires changes to the communication infrastructure. The current version of the IEC 61499 system model defines communication as segments, which are linked to devices. However, the semantics of a segment is not standardized, and the details of the various communication paradigms cannot be captured as part of the software model. Thus, there is no way to express communication behavior within the application regarding timing, style or synchronization. Moreover, due to the separation of event and data connections there might be a loss of synchronisation. In case of network communication, packets for events and data could be sent separately. This may lead to processing on obsolete data and, thus, undesired system behaviour When developers are able to specify communication across device boundaries explicitly, they can benefit from defining the communication information within their IEC 61499 application and system model.

Contributing to the platform-independence of modeled control software, this white paper aims to support developers with a model-based approach of specifying the network. Therefore, the scope of this paper is to extend the IEC 61499 model with messages that represent the logical dataflow within a distributed application model and channels to enhance the existing network segments and define the underlying communication infrastructure. Messages can be mapped onto channels, which can be used to define extra-functional properties such as timing, Quality of Service (QoS) or bandwidth for each individual logical message. This provides additional information for further optimization strategies, automated generation of the existing communication function blocks, or automated network configuratio, which significantly reduces the number of steps that are necessary for re-configuring the hardware [1]. Thereby the proposed extension of messages and channels significantly enhances the existing modelling approach of IEC 61499, supports platform-independence, and provides the basis for further automation and optimization tasks. This work is accompanied by tool support as part of the open-source Eclipse 4diac project, which serves as a reference implementation.

This work is based on the following peer-reviewed publications [2, 3, 4].

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this manual. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this manual are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

IEC 61499-1:2012, *Function Blocks — Part 1: Architecture*

IEC 61499-2:2012, *Function Blocks — Part 2: Software tool requirements*

3 Terms, definitions, and abbreviations

3.1
Message
Modelling element for network communication at application level. A message represents the logical dataflow within a distributed application model.

3.2
Channel
Channels enhance the existing network segments and define the underlying communication infrastructure, such as available time slots, bandwidth, or QoS Levels.

3.3
TDMA
The Time Division Multiple Access (TDMA), enables transmission of data from different transmitters on one network segment in specific time intervals (time slots).

3.4
TSN
The Time Sensitive Networking (TSN) standard provides several key functionalities for time synchronisation, separation of time-critical traffic from best-effort traffic, frame preemption, bandwidth reservation and more.

3.5
MQTT
MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe network protocol designed for efficient and reliable message exchange between devices over unreliable or bandwidth-constrained networks.

4 Introduction

Function blocks (FBs) support component-based design of automation systems [5, 6], which allows independent testing of each component and comes with many more advantages for formal verification procedures. In executable applications, FBs are mapped to device resources. The mapping specifies, for each FB, in which resource the FB will be executed. Mapped FBs also

appear within the resource view that corresponds to the target resource. To specify communication across device boundaries, developers can map two connected FBs onto different devices. Additionally, the developer needs to specify how data are transferred by adding communication FBs (e.g., a publish FB and a subscribe FB). Communication FBs are based on SIFBs, which specify sending data to or receiving data from another device.

Within the system model network segments represent a visual element, but does not specify any semantics of segments or links. In 4diac IDE we currently consider a segment instance as an element with an instance name, a reference to its segment type, and configuration-specific parameters.

However, this information does not affect the run-time behaviour. Thus, communication across device boundaries is only included in the IEC 61499 model as interfaces. Developers have to configure the FB (types) to implement their distributed system using various communication protocols.

IEC 61499 has several (intentional) semantic loopholes and does not offer a fixed and deterministic execution semantic [7, 8]. Accordingly, it allows different interpretations of the modelled application [9]. The execution behaviour might vary based on the utilised runtime and there is limited support for hard real-time applications. Several approaches aim to provide solutions and offer a deterministic runtime [10, 11, 12, 13] and formalisation of function blocks [7]. The work of Insaurralde [14] proposes a concept for real-time scheduling utilising the Time Triggered Protocol (TTP) for IEC 61499. While the concept is applied in the context of avionics, it is similarly applicable for industrial use cases. The existing dedicated FBs for sending (publish) and receiving (subscribe) are not enough to represent the underlying implementation in IEC 61499. Accordingly, the concept proposed in [14] does not represent the data transmission process in the application model explicitly. For example, this means that it cannot be included within scheduling approaches. This becomes an issue for several reasons:

4.1 Platform Independence

A core value for IEC 61499 systems is platform-independence of the application model. The application is supposed to be adaptable to new hardware configurations, which also requires changes to the network structure in case of distributed applications. Currently, the software components (existing communication FBs) that handle the communication have to set the parameters for a specific hardware setup, such as device IPs. Thus, if developers re-configure their hardware, they also have to adapt the parameters within the software. A separation of the logical data transmission within the software application from the necessary information for the physical communication infrastructure in the system model would support the standards platform independence. Without reflecting the separation of logical data transmission in terms of semantic and physical configuration, platform independence is not given. Within this work, we define the novel modelling elements for the logical (messages) and physical (channels) data transmission. As an example, we use the modelling elements to utilise the communication protocol TSN. However, other communication protocols can be utilised similarly. The example serves as a reference and we attached some use case scenarios.

4.2 Loss of Synchronisation

IEC 61499 separates the event and data connections allowing the event-based execution of applications. However, this might lead to a loss of synchronisation between the connections when there is non-local communication. It is not clear whether the packets for the event and corresponding data are sent simultaneously. Consequently, it remains uncertain if the data are

already updated by the time an event triggers the execution of an FB. Moreover, packet loss might occur or different routing could change the packet order. This could lead to undesired behaviour of the application. This particularly becomes an issue in real-time applications. The introduction of the message FB allows explicit modelling of communication. While an adapter FB can be utilised for a similar purpose to structure complex applications, the message FB has additional benefits especially for real-time applications.

4.3 Strictly Timed Control Systems

Considering non-local communication particularly matters when developing control systems that need to adhere to real-time constraints or that contain time-critical traffic. When transmitting data across deterministic networks, developers need to explicitly specify which data are time-critical and need to be sent within a given time frame or with a certain priority. This is currently ignored within IEC 61499 models although it potentially has a huge impact on the application's behaviour not only influencing the control performance but may even lead to unstable dynamics. Mainly, explicitly modelling network communication allows specification of communication requirements, such as communication timing. TDMA-based protocols (e.g. TSN) allow a certain time window to be dedicated to the permissible transmission of data. Then, a message FB can be mapped onto a corresponding time slot and define exactly when the data packet has to be transmitted. Consequently, this does not only influence the timing behaviour of communication, but also requires a proper specification of which data are allowed to be transmitted during which traffic window. Through message FBs, scheduling network communication at application-level becomes possible, enabling the use of automatic scheduling approaches and timing verification techniques to ensure the correct timing behaviour of the complete distributed control application.

4.4 Missing Formalisation

To date, the delay for transmitting data across device edges is not considered in verification techniques for IEC 61499, because the actual data transmission over wire that takes place in between the interfaces, is not represented in IEC 61499 as a formal model. Therefore, there is no possibility to perform formal verification of the full distributed control system that does not leave out this crucial part of data transmission.

Drozdov et al. [15] take the formal model of IEC 61499 based architectures a step further to support time-aware computations. According to their work, it is necessary to flatten the hierarchical architecture of the IEC 61499 model, which comes with certain restrictions. This allows the introduction of event timestamps that enable checking timing guarantees and implementing robust time-aware behaviour. However, their work leaves out a concept for network communication. The challenges of the IEC 61499's event-triggered mechanism for distributed real-time application with focus on communication was already addressed in 2004 by Schwab et al. [16] providing the basis for integrating communication related code within the application specification. An IEC 61499-based solution for reliable communication proposed by Atmojo et al. [17] suggests an extension for communication based on SystemJ channels. In contrast to this work, they aim to replace the present communication FBs (e.g., publish FB and subscribe FB) with an output and input channel port. Pérez et al. [18] introduce the concept of a communication channel to allow automated reconfiguration to allow responses to process changes. The communication channel introduces a new abstraction layer for communication and is used to define complex and flexible systems. Similar to this work, their concept can be used to synchronise data and event streams. On the contrary, they do not provide a formal description of the network and do not consider possible communication protocols and a systematic mapping strategy for mapping logical communication to available physical communication resources. An interesting

approach by Lednicki et al. [19] generates a communication model based on a given application and system model. The communication model allows selection of available communication media and generates communication FBs according to the provided information. Accordingly, similar to this thesis, Lednicki et al. see the need for specifying communication explicitly as well. However, their approach is focused on automated generation of protocol specific communication FBs that are already present for specific devices rather than allowing a systematic mapping of communication over wire to a specific available communication resource of a certain communication protocol.

5 IEC 61499 Modelling Extension for Network Communication

The integration of a model-based specification for (time-critical) networks requires an extension of both the IEC 61499 system and application models through the incorporation of new language elements. This extension of the standard is not only a pragmatic response to the evolving landscape of networked systems but also a critical step in enhancing the models' expressiveness and applicability especially in time-sensitive contexts. Moreover, this extension introduces the vital need for a refined mapping mechanism of logical data transmission to physical system architecture. In particular, distinct communication elements within the application model need to be mapped to available communication resources of the system model that can ensure the application requirements. Within the scope of this white paper, a communication mapping is introduced to allocate messages to their respective communication mechanism. This process draws an analogy to the mapping technique employed in IEC 61499, where FBs are systematically assigned to their corresponding execution resource. The parallel mapping approach ensures a structured and coherent allocation framework for both, common FBs and messages, within the system architecture. In turn, this requires a detailed exploration of the modelling specifications that serve as foundational guideline and prerequisites:

a) The available communication mechanisms within the network segment.

b) The messages transmitted through the network.

c) The selected communication mechanism for each message.

The subsequent clauses provide a detailed explanation of the individual elements within each IEC 61499 model. The consecutive communication mapping framework showcases the significance within the broader context of this work.

5.1 Modelling Element at System Level: Channel

A segment, within the context of this white paper, serves as a representation of the communication infrastructure connecting two or more devices. A communication resource is supposed to encapsulate the intricacies of the physical transmission of data over wire. We advocate for the incorporation of the technical parameters governing network communication as an integral aspect of the segment. This conceptual integration is realised through the introduction of a new modelling element at system level: a channels. Each channels is designed to represent the physical transmission of data over a network connection and facilitates the modelling of a communication protocol as an intrinsic part of the network segment. Consequently, a network segment possesses the potential to encompass multiple channels. The variability in the communication pattern dictates the technical parameters associated with the communication protocol, which may define cycle times, time slots of specific durations, or designated frequency ranges. The necessary parameters for defining a communication protocol should be stored as part of the

segment type definition. This way the modelling framework captures diverse technical intricacies inherent in the network segment structure.

By enabling communication configuration within a network segment, developers gain the ability to specify crucial parameters. In this case, the parameters are the overall cycle time of the schedule, the number of time slots, and the duration of each individual time slot. These specifications constitute essential information required to set up a schedule, which is necessary to determine at which point in time a message is sent. It is noteworthy that, in the definition of this segment type, developers are required to explicitly define the overall cycle time. This definition, while essential in certain configurations, is typically unnecessary for configuration of TSN setups where the cycle time can be deduced from the sum of the durations of all time slots. To enhance flexibility in schedule design, the inclusion of the overall cycle time as a configurable variable is recommended. This approach allows for the accommodation of further network devices beyond the scope of the IEC 61499 model. Thereby, the additional devices are enabled to communicate during unallocated time periods within the same infrastructure. Within this context, the term time slots is interchangeably referred to as channels. Following the definition of the segment type, it can be instantiated in the system configuration any number of times. Each instance is parameterised based on the variables stipulated in the segment type.

Ensuring that the sum of all durations does not surpass the prescribed overall cycle time is essential. However, it is not only advisable, but crucial, to allocate a buffer within the schedule. This buffer serves a dual purpose: it provides a safety margin for potential traffic outside the designated application and also establishes synchronisation of the execution cycle of the application and the network cycle. The buffer can be represented by one or more channels. This deliberate inclusion of buffer channels enhances the overall flexibility of schedule design, creating an adaptive framework where devices beyond the scope of the IEC 61499 model can communicate seamlessly during unallocated time intervals. More importantly, it offers a way to align the computational execution cycle with the network cycle. This way a synchronised and harmonious interaction between computational processes and network communication within the overall system architecture is ensured.

5.2 Modelling Element at Application Level: Message Function Block

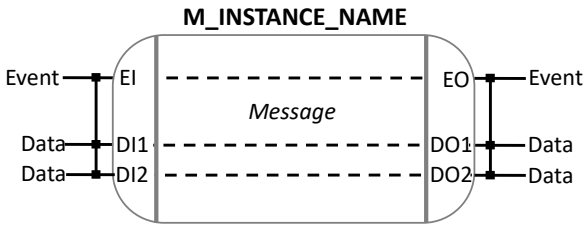
As previously outlined, the IEC 61499 application model lacks explicit representation of which data and events constitute a singular communication transaction. To address this limitation, this white paper extends the application model and introduces the novel communication FB message. This new component replaces existing symbols that represent network communication, such as the dashed directed edge in the 4diac IDE. Thus, a message serves as an abstract representation of network communication, delineating the logical transmission of data from one device to another. To incorporate this functionality, developers have to explicitly insert a message FB whenever communication occurs via one of the defined channels.

Figure 1a depicts the interfaces of a message, which has a mathematical notation similar to typical FBs as defined by Dubinin et al. [7].

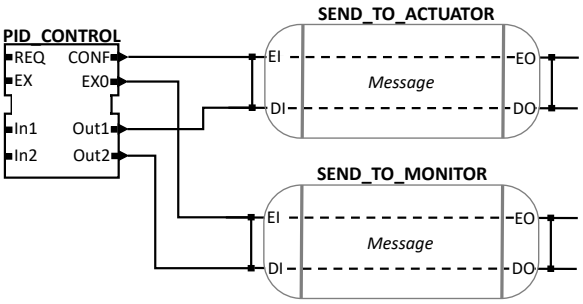
Definition 1 (Message FB) *Its interface is determined by the tuple (EI, EO, DI, DO, IW, OW) with exactly one scalar event at the input and output (EI and EO). These refer to the incoming transmission request event and transmission confirm event. A message has a set of data inputs ($DI = di_1, di_2, ..., di_j$) and an according number of data outputs ($DO = do_1, do_2, ..., do_j$). The event ensures that the transmission is synchronised, so that the event and data transmission only takes place simultaneously. As a mathematical notation, this is described as WITH-(event data) associations. For a set of inputs this is described as $IW \subseteq EI \times DI$, and for outputs the*

notation is $OW \subseteq EO \times DO$.

Remark 1 *The scalar event input EI triggers the transmission of the data input set DI.*



(a) Interface of a message. In contrast to data inputs and corresponding outputs there is only one event input and output. The WITH-association connects all data inputs with the event input and all data outputs with the event output.



(b) Context dependent modelling of separate messages by selecting a set of one event & one or more data streams.

Figure 1: A new communication function block for IEC 61499: The message.

This definition allows the utilisation of the new modelling element as illustrated in Figure 1b, where distinct event/data combinations originating from a single FB (PID_CONTROL) are individually separated and forwarded through their respective message instances, namely, SEND-_TO_ACTUATOR and SEND_TO_MONITOR).

Unlike conventional function blocks, the $WITH$-associations are obligatory, demanding that all data inputs and outputs be linked to the corresponding event input or output. Accordingly, whenever an event is triggered, some value has been updated. As long as no value has been published via the connection, the initial value of the input data pin applies. This means when a value has been published for the first time this value holds. and if the target FB is executed before the source FB, then the target FB simply takes the initial value that it has defined at the input which might be (1) defined in the instance, or if there is no initial value then (2) in the type, or if there is also no value it will take (3) the default value of the data type. The specifics of this mechanism are highly dependable on the runtime and, thus, require further investigation (refer to [20]). This results in the creation of a message encapsulating both event and data information. It is possible to define further message types to differentiate them in terms of other semantics, such as number of data to be transmitted or QoS level in case of the MQTT protocol. Notably, data transmitted through the message occurs exclusively upon the triggering of a transmission request event. Consequently, beyond its role in specifying time-critical network communication, messages also serve as synchronisation points. To fulfil this role, the message FB's interface is intentionally constrained to a single event pin. The n input data pins are associated with this input event, while the n output data pins are correspondingly aligned with the output event. This design ensures that all associated data are transmitted across the network simultaneously with the event, enhancing the message's functionality as a synchronisation point within the distributed system. Consequently, the number of data inputs/outputs define the type of a message. For example, a message FB with two data inputs/outputs is referred to as message type MESSAGE_2.

Moreover, the specification of messages within the application contributes the predictability of network communication. This is achieved by modelling the logical communication through messages, the physical communication as a deterministic communication protocol in the system model, and the strategic mapping of messages to designated communication resources (channels). Thereby, the overall modelling of communication ensures that critical data updates may not be

overlooked. In essence, the integration of messages enriches the overall communication framework, contributing to a more structured and reliable architecture aligned with typical IEC 61499 application and system modelling. Importantly, developers retain the flexibility to opt-out of incorporating messages into a distributed system. Instead, messages are intentionally designed to serve as a tool for specifying particular aspects of the application's network communication that necessitate special care, primarily those characterised as time-critical requirements. This strategic design choice provides developers with the flexibility to designate specific segments of the application's network communication as time-critical. In scenarios where a message is integrated to manage specific network communication aspects within an application, the remainder of the application's network communication operates on a best-effort basis.

5.3 Transmission Model for Messages

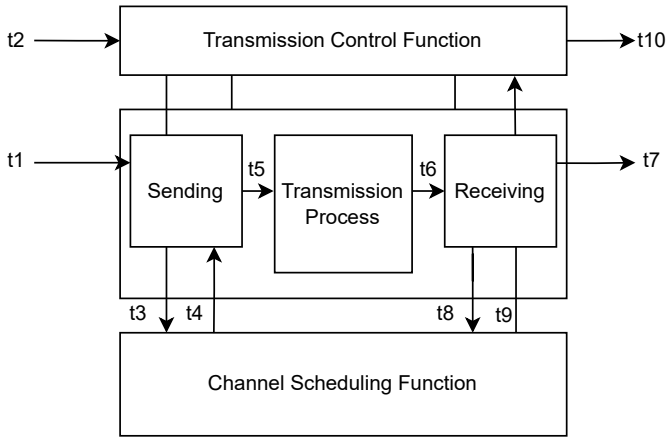


Figure 2: Transmission Model.

NOTE This figure is illustrative only. The graphical representation is not normative.

Figure 2 represents the transmission model of a message corresponding to the execution model defined for typical FBs. Thereby, we highlight the difference in execution behaviour. For example, in contrast to FBs the message FBs have to take into consideration the time delay for sending and receiving a data packet. Moreover, the message transmission behaviour integrates a scheduling function, which refers to the communication protocol modelled using channels in the system models' network segment. The overall process can be described as follows. Sending data packets for messages is invoked by the transmission control portion of the block instance in response to events at event inputs. This invocation takes the form of a request to the scheduling function of the associated resource to schedule the transmission of a data packet. Upon completion of transmission of a data packet, the transmission control generates an event at the event output. Events at event inputs are provided by connection to event outputs of other function block instances. Events at these event outputs may be generated by transmission control as described above, or by the communication mapping, process mapping, scheduling, or other functional capability of the resource.

t_1: relevant input variable values (i.e., those associated with the event input by the WITH qualifier) are made available;

t_2: the event at the event input occurs;

t_3: the transmission control function notifies the channel scheduling mechanism to schedule the data packet for sending;

t_4: sending begins at the sending device;

t_5: the actual transmission process starts

t_6: the transmission process completes sending over wire and provides output variables to the receiving device;

t_7: receiving completes and provides output variables associated with the event output by the WITH qualifier;

t_8: the channel scheduling function is notified that the overall transmission process has ended;

t_9: the scheduling function invokes the transmission control function;

t_{10}: the transmission control function signals an event at the event output.

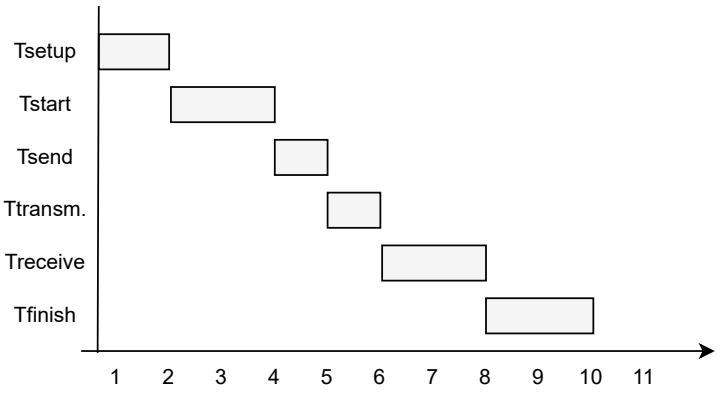


Figure 3: Transmission Timing.

NOTE The axis labels 1,2,... in the above figure correspond to the times t1, t2,... of Figure 2

The significant timing delays in this case which are of interest in application design are:

$T_{setup} = t_2 - t_1$

$T_{start} = t_4 - t_2$ (time from event at event input to beginning of sending process)

$T_{send} = t_5 - t_4$ (send data packet through network stack of the sending device)

$T_{transmission} = t_6 - t_5$ (time for transmission over the network)

$T_{receive} = t_8 - t_6$ (receive data packet through network stack of the receiving device)

$T_{finish} = t_{10} - t_8$ (time from end of receiving process to event at event output)

5.4 Mapping Model

The introduced extension allows the mapping of messages onto channels (see Figure 4). This mapping process is critical, because it precisely dictates the temporal aspects of message transmission over the network. The mapping mechanism serves as a tool for specifying which messages are slated for transmission during specific and well-defined time slots.

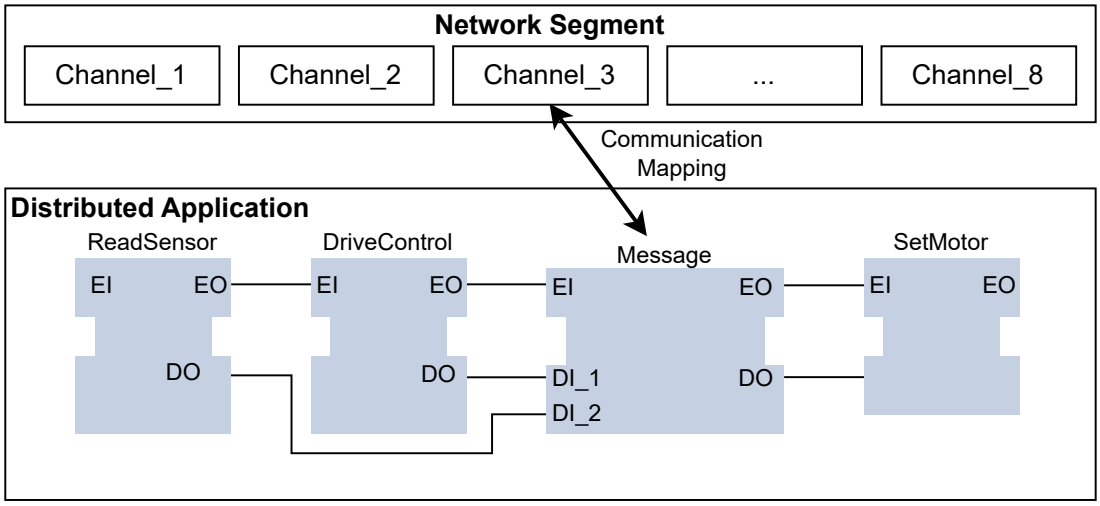


Figure 4: IEC 61499 system & application model of a distributed application using an IEEE 802.1Qbv network as an example enhanced with communication model elements message and channels for data transmission across device boundaries.

The process of mapping messages onto channels requires the developer to decide about when a message should be transmitted. Here, the concept allows for direct and manual specification. However, the concept of messages also enables the application of common task scheduling strategies for network communication. Consequently, the chosen scheduling strategy determines the specific order and timing of messages sent across the network. While this work assumes that message transmission is scheduled statically and offline by the developer during system development, the concept is adaptable to dynamic scheduling techniques. To use channels for specifying different communication protocols, it needs to be derived what minimal information is essential for the protocols' configuration, such as specifying the available bandwidth, priority, or QoS level. Then, channels can be used as modelling elements to distinguish between different parameter values or create a schedule and serve as a mapping target for messages.

In practice, the developer establishes a mapping aligned with the control tasks' requirements. The order of FBs is explicitly specified by the developer within the IEC 61499 application model. Accordingly, the focus is on finding a feasible schedule for the message FBs. During the message mapping process, the entire task set (comprising each (message) FB of the system) is considered and analysed to determine the required order for transmitting data packets as represented in Listing 1. All FBs, including the messages, are scheduled accordingly. Thereby, the resulting set of sorted message (O_{msg}) inherits corresponding timing information. The concept proposed in this work supports utilisation of established scheduling algorithms. The network schedule can be defined based on the order and duration of FBs. Consequently, a channel for sending a message is introduced whenever data need to be transmitted and buffer channels are integrated into the network specification serving as a synchronisation mechanism between the network and execution cycles. Devices outside the scope of the IEC 61499 model can communicate during unallocated time slots (buffer channels). This provides more flexibility in event-based systems, while still adhereing to real-time requirements on a reliable time-triggered basis that can be enforced for the time-critical parts of an application. However, this requires knowledge of the timing properties for an application that could be integrated on the basis of timing analysis techniques. To be able to integrate the available timing information, IEC 61499 applications and FBs require a formal extension to integrate timing properties.

```
1  Input: Set of messages msg_i
2         Set of channels cc_j
3  Output: Set of scheduled messages O_msg
4
5  i, j ← 0
6  O_msg ← {NULL}
7  for each cc_j do
8          if cc_j = NULL
9                  cc_j ← msg_i
10                 i ← i + 1
11         end if
12 end for
```

Listing 1: Message analysis and scheduling.

Beyond its temporal dimension, the mapping mechanism provides an advantage by offering a comprehensive overview of the messages intricately associated with a particular channel. This additional layer is especially beneficial when accessing the resource view of a segment. Here, a list of messages could represent all messages that are statically mapped onto a specific channel, providing a detailed and accessible representation of the communication structure within the segment. Furthermore, the mapping capability is not only about visualisation, it also enables automated processes by including mapping information within the broader system and application models. Utilising the respective information the communication FBs, such as publish and subscribe, could be generated automatically. The generated blocks, representing the communication interface at the devices, could then be visualised. This enhances the clarity of communication representation and the overall understanding and management of the communication architecture within the modelling environment.

6 Proof-of-Concept Implementation in Eclipse 4diac

While the general concept can be universally applied to any IEC 61499-implementation, this work goes a step further by presenting a proof-of-concept implementation tailored for the open-source project Eclipse 4diac. The publicly available code serves as a reference implementation for the approach.[2)]

The integration of these novel modelling elements is realised within the Eclipse Modelling Framework (EMF). The meta-model depicted in Figure 5 provides a comprehensive view of the extended communication infrastructure. Notably, the IEC 61499 meta-model incorporates generic communication elements, enhancing its flexibility. In the meta-model, each `Segment` possesses the capability to reference an object of type `CommunicationConfiguration`. This object, while optionally set to `null` if no configuration parameters are specified, encapsulates all the configuration details required for communication within the segment. A fundamental addition to the meta-model is the introduction of message as an abstract representation of unidirectional communication between two application components. Similar to other FBs, the message can be instantiated within the application and subsequently mapped to a `CommunicationMapping-Target`. The IEC 61499 meta-model does not need any specific details about the available channels. Instead, a `CommunicationConfiguration` provides an abstract method for returning the relevant `MappingTargets`, highlighting the modular and adaptable nature of this communication infrastructure extension.

[2)] `git.eclipse.org/c/4diac/org.eclipse.4diac.ide.git/tree/plugins?h=develop`

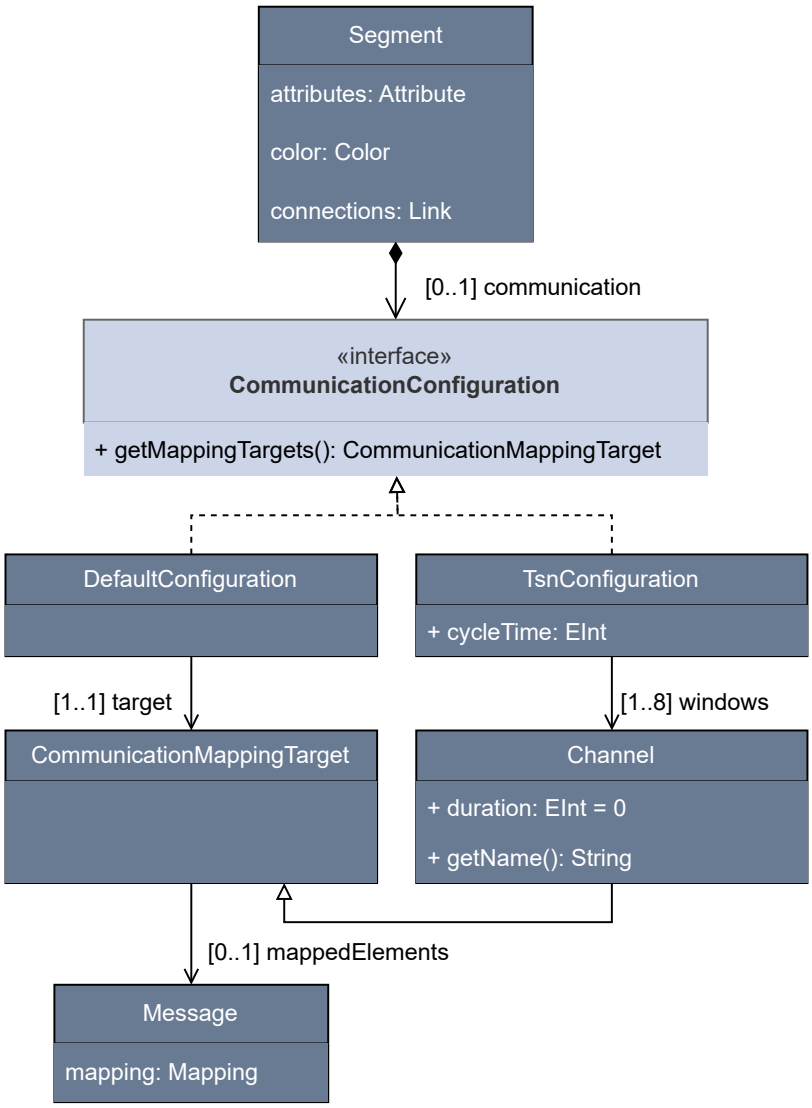


Figure 5: Meta-model of the communication infrastructure.

Within this communication meta-model, the specific configurations that will be integrated in the tool have been implemented. This white paper offers two exemplary implementations to illustrate the versatility of the approach:

a) The `DefaultConfiguration` defines a `Segment` featuring a single communication slot named `target`, specifically tailored for best-effort traffic scenarios.

b) The `TsnConfiguration` serves as a realisation of the standard outlined in clause **??**, providing a concrete implementation of the proposed concepts and principles in the communication meta-model. Within this configuration, there are the parameters cycleTime, along with a list of up to 8 time windows denoted as a `Channel` and each with its specified duration. The limitation of 8 `Channels` could be easily adapted, when there are respective changes to the TSN standard. Each of these channels, functioning as a `CommunicationMappingTarget`, has the flexibility to accommodate any number of mapped messages. It is important to note that the cycle time must be equal to or greater than the sum of durations of the specified time slots, ensuring proper temporal alignment within the configuration.

Both implementations have been integrated into 4diac IDE as plugins that specify a so-called extension point. In the Eclipse platform, extension points serve as a mechanism for facilitating customization within Eclipse-based applications, such as 4diac IDE. This communication extension point enables new plug-ins to provide detailed information. This information is handled by the core IDE through a specified interface.

The additional views implemented as an extension to the 4diac IDE are designed to be indifferent to the underlying communication protocol. This design choice ensures that users can introduce new configurations for customisation without necessitating modifications to the core IDE. The process involves three essential parts:

a) Segment Type Specification: A segment type, defining the configuration parameters, serves as the foundational element.

b) Extension of the Communication Meta-Model: The communication meta-model needs to be extended to accommodate the new configuration type, integrating it into the existing framework.

c) Implementation of an Interface: `CommunicationConfigurationDetails` is provided as an interface, necessitating implementation in a class that integrates all essential information to the IDE. This class, crucial for the functionality, is then registered through the designated extension point, ensuring its recognition and utilisation within the IDE.

Some minor adjustments to 4diac IDE were required, as the tool did not provide complete support for the IEC 61499 standard. These adaptations include augmenting the segment meta-model with a list of variables to ensure compatibility with the standard.

7 Application Guidelines

7.1 Categorization for Specification

Communication protocols can be divided into categories based on where in the communication stack they operate and what they influence. This can help clarify whether a protocol applies to configuring network segments (system-level concerns), communication FBs (transport layer concerns), or logical messages (application-level concerns). However, most protocols have cross-layer dependencies and impact commmunication aspects at different layers. Therefore, the goal of the following categorization is to serve as a guide to decide which aspects of a communication protocol should be modeled with the messages and channels of the novel communication model elements.

Communication protocols that mostly influence the system level are protocols that operate at the Network and Link Layer (Layers 2 and 3 of the OSI reference model). There is a multitude of Ethernet-based variants (e.g., Ethernet/IP, Profinet, Modbus) that enhance the specific performance parameters or reliability. The Time-Sensitive Networking (TSN) standard is a collection of sub-standards that aims to unify the various Ethernet-based protocols to enable deterministic communication and precise timing. Other examples are wireless network protocols such as Wi-Fi, ZigBee, or BLE. Consequently, the goal of enhancing the network segment in the system model is, to be able to represent the different kind of communication protocols. This has to include modelling the topology, define extra-functional characteristics such as timing, defining available bandwidth. In particular, extra-functional characteristics could be modelled by partitioning a network segment into a number of channel modelling elements.

At the Transport Layer, communication protocols ensure reliable or efficient data delivery between endpoints. They typically influence both network configuration and application behaviour. Some examples are the TCP (Transmission Control Protocol) for reliable, connection-oriented communication, UDP (User Datagram Protocol) representing unreliable, connectionless communication with low overhead, or QUIC as a modern transport protocol for web applications that combines reliability and speed. The existing Communication FBs, such as publish/subscribe or client/server, focus on this aspect of communication. For 4diac IDE, there exist some approaches to enhance the Commmunication FBs by adding the additional information needed for other protocols as a string. However, it is not possible to capture the underlying topology or define extra-functional properties for the communication infrastructure.

Application Layer Protocols often encapsulate domain-specific functionality and are implemented in software. Some examples are MQTT (Message Queuing Telemetry Transport), a lightweight publish/subscribe protocol, CoAP (Constrained Application Protocol), which is designed for constrained devices and uses RESTful communication over UDP, or HTTP/HTTPS as a foundation for web communication. In general, the focus at application layer lies on the semantics of data and specific interactions. For instance, MQTT defines the interaction between components by defining the data payload and its semantics and additionally, it allows specifying three possible Quality of Service (QoS) level for each individual data packet. While the content of the data packet should be modelled in the application view, the QoS level specification should be part of the system model's network segment. This way, a data packet (modelled as a Message FB) at application level can be mapped to a specific QoS level within the system model.

Table 1 provides an overview on different criteria, which represents a basis to decide whether the protocol applies to network infrastructure (network-segment-level), application functionality (message-level) or could be represented by the existing communication FBs. This ensures that the overall network model accurately reflects communication needs.

Table 1: Decision Basis: Network Segment vs. Communication FBs vs. Logical Message

	Network Segment	**Communication FBs**	**Logical Message FB**
Focus	Topology, bandwidth, timing	Communication patterns	Semantics, data payloads
Layer	Physical, data link, network	Transport	Application
Configuration Scope	Components of the network	Communication mechanism	Data flow

7.2 Usage Guidelines

As a generalization, we define the following steps to decide on where to configure the different protocols:

a) Determine communication purpose: Is the protocol influencing network behavior or defining message structures and semantics? Either focus on network configuration or logical message definition within the application, respectively.

b) Check for cross-layer dependencies: If the protocol affects multiple layers address both network segment configuration using channels and logical messages.

c) Align with system model goals: Network segment configuration to reflect the topology, timing, bandwidth management, or deterministic behavior and message definition for semantics, application-layer policies, or data structure definitions.

d) Mapping of logical messages: Depending on the system topology it is necessary to map

messages to either network segments or specific channels.

8 Summary

In summary, the newly introduced extension stands as an enhancement to the network communication modelling. Central to this enhancement are the concepts of messages and channels, which in turn allow explicit mappings to associate messages with designated channels within the network infrastructure. As an example, this functionality offers control over the precise timing of message transmission, thereby contributing to a more refined and predictable communication framework. The visual representation within 4diac IDE adds an additional layer for clarity and structure within a given communication segment. Beyond the timing schedule specification for network communication via channels, the extension introduces the message concept as a synchronisation point that alignes event and data streams. The incorporation of channels enhances the coordination within the network and, additionally, allows buffer channels. Buffer channels enable the synchronisation of the execution and network cycle. This dual synchronisation approach further refines the predictability and reliability of network communication. Moreover, the novel modelling elements represent a fundamental advancement in the formal definition of an IEC 61499 application. The formalisation serves as a foundation to perform verification and validation techniques for the overall distributed control application. A formalised structure facilitates systematic assessment of system properties to ensure robustness, reliability, and adherence to specified requirements. This, in turn, enhances the overall quality and dependability of the developed systems. The synergy between novel modelling elements and an advanced mapping mechanism elevates the formal definition of IEC 61499 applications and establishes a foundation for automated workflows. As an example, communication function blocks (publish and subscribe) could be generated based on the mapping information. Moreover, the network information can be used for automated network configuration. This automation streamlines the development process and reduces the potential for manual errors.

The key features of messages and channels can be summarised as follows:

a) IEC 61499 Extension: This extension allows specifying communication mechanisms within the network through channels. It enables message specification and mapping onto channels. Thereby, selecting the communication mechanism for each designated message.

b) Timing Specification: The mapping process precisely determines when a message is transmitted over the network. This way, developers can explicitly control the timing of network communication.

c) Synchronisation: Messages are utilised as a synchronisation point for event and data streams. Additionally, it introduces the option to incorporate buffer channels into the network specification, which serves as a synchronisation mechanism between network and execution cycles.

d) Formalisation: Explicit formal specification of network communication and timing behaviour in IEC 61499.

e) Visualisation: Mapping of messages to channels enables a clear overview of how messages will be transmitted. The extension enhances the visualisation within the resource view of a segment and within the application model.

f) Verification & Validation: The formalised structure enables the use of verification and val-

idation techniques for complete distributed applications including network communication to ensure robustness, reliability, and compliance with specified requirements.

g) Design Automation: The additional information can be utilised to automatically generate communication function blocks (e.g., publish and subscribe). Moreover, the included network parameters enable automated configuration of the network infrastructure. These aspects significantly streamline the development process.

Annex A
(normative)
Extensions to IEC 61499-1 Annex A

The syntax defined can be used for the textual specification of message types according to the rules given in Clause 5 of this standard.

```
/* Modelling Elements at Application Level */
message_type_declaration::=
    'MESSAGE' message_type_name
    message_type_list
    message_interface_list
    'END_MESSAGE'

message_interface_list::=
    [event_input_list]
    [event_output_list]
    [input_variable_list]
    [output_variable_list]
    <as defined in IEC 61499-1>

message_type_list::=
    'MESSAGE_TYPES' {message_type_name ';'}
    'END_MESSAGE_TYPES'

fb_instance_list::= 'FBS'
{fb_instance_definition ';'
 message_instance_definition ';'}
'END_FBS'
<In context of subapplications, messages can be treated like basic FBs>

message_instance_definition::= message_instance_name ':' message_type_name
    [parameters]

message_type_name::= identifier;
message_instance_name::= identifier

/* Modelling Elements at System Level */
segment_configuration::=
    'SEGMENT' segment_name ':' segment_type_name [parameters]
    [channel_type_list]
    {channel_configuration}
    [channel_instance_list]
    [segment_config_connection_list]
    'END_SEGMENT'

channel_type_list::= 'CHANNEL_TYPES'
    {channel_type_name ';'}
    'END_CHANNEL_TYPES'

channel_configuration::=
    'CHANNEL' channel_instance_name ':' channel_type_name [parameters]
    [message_type_list]
    [message_instance_list]
    [segment_config_connection_list]
    'END_CHANNEL'

channel_instance::=
    'CHANNEL' channel_instance_name ':' channel_type_name
```

```
    [message_instance_list]
    [segment_config_connection_list]
    'END_RESOURCE'

channel_type_specification::= 'CHANNEL_TYPE' channel_type_name
    [input_variable_list]
    [message_type_list] <if not given, defined by message instances>
    [message_instance_list]
    segment_config_connection_list
    'END_CHANNEL_TYPE'


message_instance_reference::= [app_hierarchy_name] message_instance_name

message_mapping::= message_instance_reference 'ON'
   message_channel_reference ';'

message_channel_reference::= channel_hierarchy ['.' message_instance_name]
    <When the optional element ['.' message_instance_name] is not given,
   the
    instance name of the message in the channel is the same as its instance
    name
    in the corresponding message_instance_reference of the mapping.>

channel_hierarchy::= segment_name ['.'  channel_instance_name]

segment_config_connection_list::=
    [segment_config_event_conn_list]
    [segment_config_data_conn_list_in]
    [segment_config_data_conn_list_out]

segment_config_event_conn_list::= 'EVENT_CONNECTIONS'
    {segment_config_event_conn_in}
    {segment_config_event_conn_out}
    'END_CONNECTIONS'

segment_config_event_conn_in::= fb_instance_name '.' event_output_name
    'TO' message_instance_name '.' event_input_name ';'

segment_config_event_conn_out::= message_instance_name '.'
   event_output_name
    'TO' fb_instance_name '.' event_input_name ';'

segment_config_data_conn_list::= 'DATA_CONNECTIONS'
    {segment_config_data_conn_in}
    {segment_config_data_conn_out}
   'END_CONNECTIONS'

segment_config_data_conn_in::=
    (fb_instance_name '.' output_variable_name | input_variable_name)
    'TO' (message_instance_name | channel_instance_name)'.'
   input_variable_name ';'
    <channel_instance_name only applies to connections within segment_type
   or
    segment_configuration declarations>

segment_config_data_conn_out::=
    (message_instance_name '.' output_variable_name | input_variable_name)
    'TO' (fb_instance_name | channel_instance_name)'.' input_variable_name
```

```
';'
 <channel_instance_name only applies to connections within segment_type
or
 segment_configuration declarations>
```

Listing 2: EBNF for Communication Modelling

Annex B
(normative)
Extensions to IEC 61499-1 Annex B

This Annex presents Document Type Definitions (DTDs) for the exchange of IEC 61499 library elements between software tools. These DTDs are defined in the syntax defined in the eXtensible Markup Language (XML) specification.

DTD Element	Library Element	Textual Syntax
MessageType	MessageTypeDeclaration	message_type_declaration
ChannelType	ChannelTypeDeclaration	channel_type_declaration

Table B.1: Document type definition (DTD) elements

```
/*Example for providing a communication protocol via the extension point:
   TSN configuration. */
<communicationProtocol
    class="org.eclipse.fordiac...TsnDetails"
    id="EthernetTSN"
    label="TSN" />

/*Instance definition for two distinct message FBs. */
    <Application Name="App" Comment="">
  ...
  <FB Name="SEND_TO_ACTUATOR" Type="MESSAGE_1"
      Comment="" .../>
  <FB Name="SEND_TO_MONITOR" Type="MESSAGE_2"
      Comment="" .../>
  .... <!-- connections -->
</Application>

/* Mapping of message FBs to a communication channel. */
<Mapping From="App.MsgWith1DataPin"
         To="Tsn10.ChannelP0"/>
<Mapping From="App.MsgWith2DataPins"
         To="Tsn10.ChannelP1"/>

/* Type definition for a segment supporting TSN over Ethernet. */
<SegmentType Name="EthernetTSN" Comment="">
  <Identification .../>
  <VersionInfo .../>
  <CompilerInfo/>
  <VarDeclaration Name="CycleTime" Type="TIME"
        InitialValue="T#10ms" Comment="Cycle Time"/>
  <VarDeclaration Name="ChannelP0" Type="TIME"
         InitialValue="" Comment=""/>
  <VarDeclaration Name="ChannelP1" Type="TIME"
        InitialValue="" Comment=""/>
          ...
  <VarDeclaration Name="ChannelP7" Type="TIME"
         InitialValue="" Comment=""/>
</SegmentType>


/* Instance definition for a segment supporting TDMA-based communication
   protocols, such as Time-Sensitive Networking (TSN), with four channels
   and a cycle time of 10 ms. */
```

```
<Segment Name="Tsn10" Type="EthernetTSN" Comment="" ...>
 <Attribute Name="Color" ... />
 <Parameter Name="CycleTime" Value="T#10ms"/>
 <Parameter Name="ChannelP0" Value="T#7ms"/>
 <Parameter Name="ChannelP1" Value="T#1ms"/>
 <Parameter Name="ChannelP2" Value="T#1ms"/>
 <Parameter Name="ChannelP3" Value="T#1ms"/>
</Segment>
```

Listing 3: XML for Communication Modelling

Annex C
(informative)
Application Examples

In the following sections, multiple software applications are introduced to demonstrate the utilisation of messages and channels based on the proof-of-concept implementation starting with the general modelling approach.

Afterwards, the concept is applied to a valve control as a basis to specify a wide variety of communication scenarios as benchmarks for timing analysis and evaluation of the automation process. These benchmarks contain diverse communication patterns and, therefore, ensure a broad analysis of timing behaviour within the feasibility study.

Furthermore, the second running example for joint control is put to use. Thereby, showcasing the applicability and analysing the proposed concept of message and channels, which respectively enable the systematic construction process of distributed control systems. This application is extended by introducing more communicating devices and considering the extension as part of a reconfiguration step (an important characteristic of industrial distributed control systems) to analyse scalability of the proposed concept.

Lastly, several important corner cases are evaluated by including a feedback loop within the software, thereby, creating a closed-loop system. industrial distributed control systems are typically structured as open-loop systems, since feedback can be integrated through implementation specifications depending on the use case. However, closed-loop control is a relevant aspect for systems that need to handle uncertainties and possible disturbances and, therefore, require constant adjustments.

In all scenarios, the initiation of FBs occurs in response to platform ticks, happening every 1 ms in the form of FB events along with their associated data. FB execution is limited to a maximum of 1 ms. Consequently, the deadline for FBs is set at 1 ms, assuming consistent adherence to this deadline. The mapping process for network communication specifies the utilised communication schedule. Accordingly, the TAS is created based on the presented message FBs creating channels with a duration of 1 ms for each message. Consequently, buffer channels could be equipped with different durations to establish a network cycle compatible with the application cycle. Thereby, the network and the process schedule are synchronised to create a fully deterministic application.

C.1 General Modelling Approach

This section describes the general IEC 61499 modelling approach using an example TSN configuration in 4diac IDE utilising the proposed proof-of-concept implementation as detailed in Clause 6.

Configuring each segment instance individually is necessary to support a defined number of channels. The UI displayed in the property sheet is defined through the `TsnDetails`-class provided by the extension point. The developed TSN UI is presented in Figure C.1 on the right.

Each message FB can be associated with one of the channels configured for the EthernetTSN segment. Figure C.2 illustrates the UI for mapping a message to a channel. The block is instantiated in the graphical application editor and then linked to a selected communication channel through the dialog (see Figure C.2 on the *left*). The colour of a mapped channel aligns with that of the target segment, mimicking the behaviour of FB instances and their resource (*right* Figure C.2). All messages mapped to a specific channel are represented in a dedicated

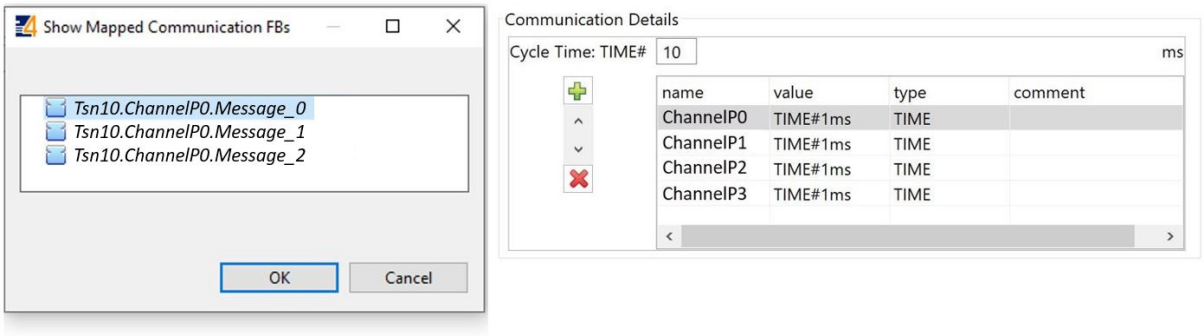


Figure C.1: Configured instance of a EthernetTSN segment, named `TSN10`, with four channels (*right*). Three messages are mapped to channel *ChannelP0* (*left dialog*).

dialogue, accessible via the communication details view (see Figure C.1). In this instance, three out of four messages were mapped to *ChannelP0*. Given the availability of four channels compared to the number of message FBs, the developer could opt to allocate a message to each channel individually.

As a message FB inherently contains information about the data and event transmitted between devices, automatically generating the corresponding communication FBs (publisher/subscriber) becomes feasible for streamlining the development process. This preventive measure ensures that developers cannot accidentally separate the transmission of an event from its associated data. Moreover, the incorporated information can be used for automated network configuration.

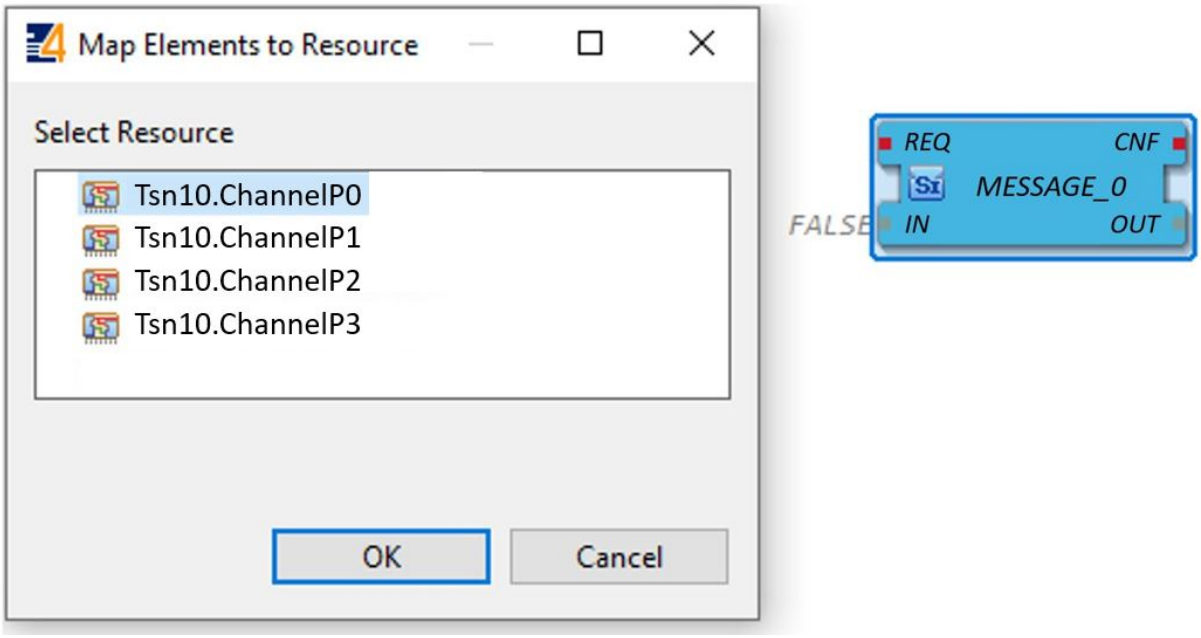


Figure C.2: Message mapped to the EthernetTSN segment. The mapping dialog allows to map a previously unmapped message, or to change the target channel.

C.2 Valve Control

The first example application for valve control consists of three FB tasks that are executed following a static scheduling policy. The functionality relies on a binary control flow where the second FB toggles a bit based on the sensor input values to decide whether to open or close a

valve.

Figure C.3 illustrates the IEC 61499 control application for valve control created with 4diac IDE as previously introduced in Clause 6, however, it is extended with time measurement FBs for further evaluation purposes. Here, a single data packet with a size of 61 bytes is transmitted a time across the network by utilising the message FB. Depending on the evaluation goal, the sending interval of the IEC 61499 application varies.

The specification assigns the first four FBs to the SENDER device and the remaining two FBs to the RECEIVER device. The devices are directly connected via Ethernet. The first FB (*E_CYCLE*) triggers the cyclic execution of the application. Upon receiving the corresponding event, the FB *READ_DATA* produces an output value (*RD*) in the form of a sine curve, simulating incoming sensor data, like from a weight sensor. The FB *DECIDE* uses the data to determine whether to open or close the valve corresponding to a desired set point generating a boolean value (*Open*) that is sent over Ethernet to the RECEIVER. *SET_CONTROL* then adjusts the actuator according to the input value *SD* to be either opened or closed.

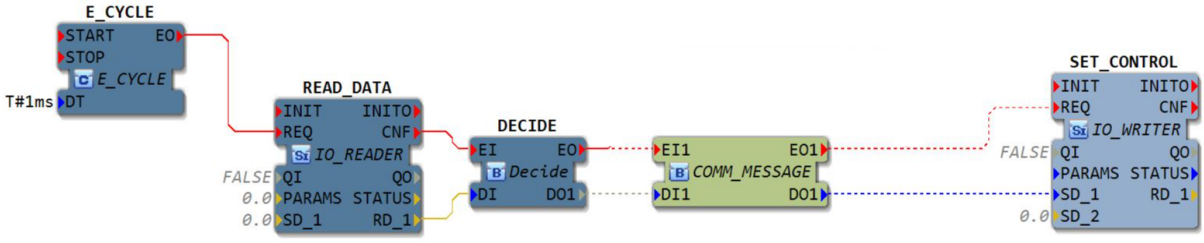


Figure C.3: IEC 61499 valve control application.

C.3 Joint Control

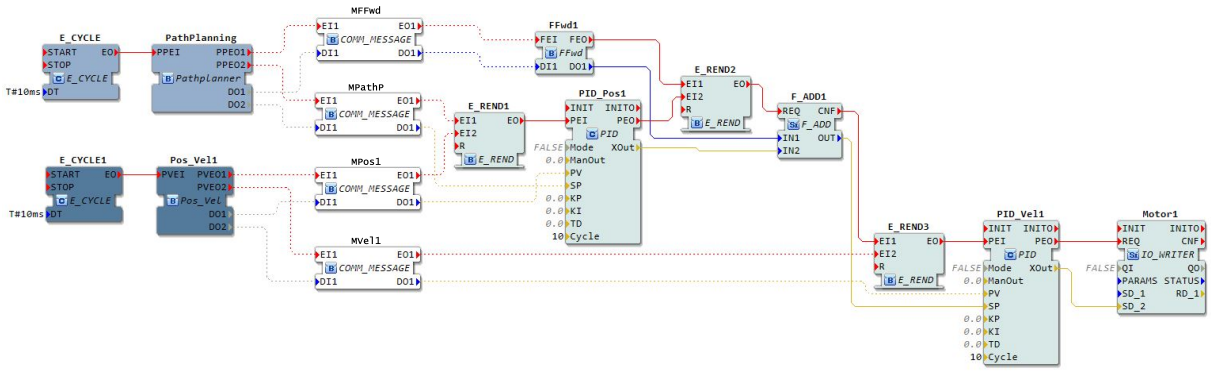


Figure C.4: Joint control application with four unmapped messages.

This second example for joint control resembles a more realistic control example for demonstrating the practical application of the proposed concepts within the tool and modelling capabilities. The control application illustrated in Figure C.4 consists of an FB network that is already mapped to three devices. The corresponding system model for the example control application is depicted in Figure C.5. The three embedded devices are connected over a common network segment. Two devices are sending data, the third device receives these data. Therefore, the two sending devices need to be triggered by an *E_Cycle* FB (here, every 10 ms). Note that the *E_Cycle* FBs are not considered within the schedule, since they are triggering the application's execution. For this application example we assume that all devices are highly synchronised based

on the PTP synchronisation mechanism resulting in the application to be triggered simultaneously on both sending devices. All non-local communication is specified using dedicated message FBs. This ensures the synchronisation between event and data stream as they are combined into a single block. The TAS schedules' overall cycle time corresponds to the application's cycle time of 10 ms. In general, creating a schedule that respects the application's timing requirements becomes more demanding the more messages developers have to consider (generating an optimal TSN schedule is an NP-hard problem [21]). All present FB tasks including message FBs serve as an input for the mapping algorithm.

As represented in Figure C.4, the first device (*PPU*) holds the FB for path planning (*PathPlanning*). The calculated path and feed forward is then transmitted to the receiving device (*MCU*). The second device (*TU*) is responsible for retrieving the current position and velocity of the joint (*Pos_Vel1*) and sends it across the network to *MCU*, which then operates on the received values. Since there is no dependency between the FBs of both devices, they can be executed at the same time (0-1 ms). Lastly, the *MCU* holds the remaining FBs. Starting with the first PID controller (*PID_Pos1*) of the cascaded control structure, the FB requires data from the *PathPlanning* FB and the recent position data from the *Pos_Vel1* FB. Assuming path planning data (*MPathP*) are sent first and, then, *MPos1*, which holds the position data, is sent afterwards. *PID_Pos1* has to wait for both received messages before starting execution. During execution, *MFFwd1* is sent for feed forward value calculation by *FFwd1*. During execution of *FFwd1*, the current velocity value *MVel1* is transmitted. Afterwards, the result from *PID_Pos1* is added to the result of *FFwd1* using an *Add1* FB. The last message, *MVel1*, is sent so that *PID_Vel1* can execute based on the received values. Finally, the result is applied to the *Motor1* FB.

Note that *E_REND* FBs are integrated into the application whenever two event streams are needed to execute the subsequent FBs. While *E_REND* FBs are necessary to unify and synchronise two event streams, assuming the application is perfectly synchronised, this in turn results in no delay generated by the *E_REND* FBs.

When message FBs are used at each of the four sending occurrences, the sending process can be executed respecting the application's timing requirements. Communication FBs (publisher and subscriber) are integrated into the application at each device whenever sending or receiving a message. The point in time for sending a message is defined by mapping it to a certain channel. Therefore, after specification of the order of messages, the next step is to specify the corresponding channels in the Ethernet segment within the system model. The segment type for TSN over Ethernet (**??**) is available in the type library. When instantiated in the graphical editor (Figure C.5), the XML of the system model is enhanced with the instance information defined in **??**.

Compliant with IEC 61499 platform independence, it is possible to change the mapping of messages. In case of *MPathP* and *MPos1* it might not make a difference. However, it does influence the application's execution and timing behaviour when mapping *MVel1* to an earlier time window than *M_Pos1* and, in turn, *PID_Pos1* has to wait the corresponding additional delay for its data. Based on the timing information, the contract specification becomes applicable and it is possible to validate the correctness of the application.

C.4 Joint Control Extended

By extending the previously described joint control application with a second joint, two additional devices (*PPU2* and *TU2*) are included within the system. The corresponding IEC 61499 application is depicted in Figure C.6.

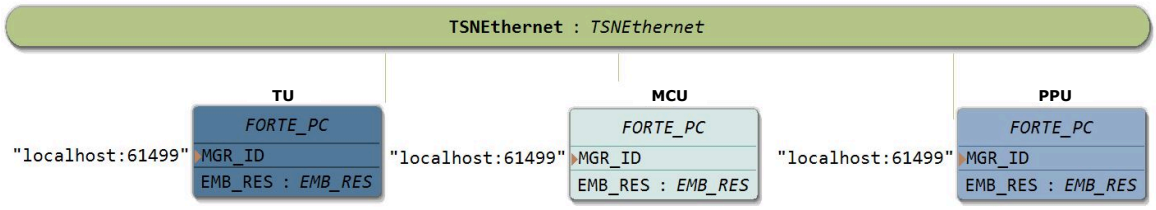


Figure C.5: IEC 61499 system configuration with an instance of the Ethernet segment for the TSN over Ethernet that connects three devices. The network (including hardware such as a switch) is abstracted as a segment.

Assuming the application extension represents a reconfiguration step in an existing system, the mapping process aims not to interfere with the already established process schedule but to adjust the network cycle as well as finding a mapping for the new components of the application. While removing a stream from the schedule does not impact any of the other streams, integrating new streams is rather complicated. Consequently, reconfiguration of existing schedules is a challenging problem. The reason is that the offsets of already scheduled streams may have to change when new streams are added to the schedule or that queues are allocated with other streams placing constraints on the achievable timing for new streams. Typically, the goal is to preserve the timing of the previous schedule as far as possible. Thus, the mapping process has to incorporate a mechanism that takes this issue into consideration.

According to the previous mapping, the initial message sent across the network contains the path planning data directed to *PID_Pos*. Given there exist one *MCU* for each of the two joints, the data can be sent to both devices simultaneously, relying on the 1:n communication paradigm. Multicast streams with more than one destination are typically considered as a tree in the form of unicast streams leading to modelling the same frame multiple times. Here, it is evident, that the concept of messages allows for modelling multicast streams within one modelling element instead of assuming all streams to be unicast. Consequently, scheduling algorithms do not have to specifically contain considerations for multicast streams and can simply rely on the modelled messages. Thereby, the first two messages and their mapping are preserved considering the synchronisation of the network with the processing cycle. Both *MCUs* can start execution upon receiving the required data.

However, to initiate the execution of *PID_Pos* on each *MCU*, the position data from the corresponding *TU* are necessary. As a consequence, the network must reserve time windows for two additional messages (containing position and velocity data of the second joint). In case there is no feasible schedule, the partitioning of the application could be revisited.

C.5 Joint Control with Feedback Loop

Extending the joint control application with a feedback loop creates a closed-loop system where the FB responsible for global path planning is updated to take the current motor speed into account, referred to as *PathPlanningFB*. This is achieved by both *Motor* FBs sending the current speed data to *PathPlanningFB*. The introduction of the *E_Rend* FB ensures that *PathPlanningFB* only starts executing when both message arrive in time. This white paper assumes that *PathPlanningFB* implements a mechanism that enforces to apply the motor's feedback only after the very first cycle (after the motor is initially set to a specific value). Additionally, it requires receiving both feedback messages as well as the *E_Cycle* event occurrence every 10 ms. Figure C.7 visualises the extended application for control of two joints with feedback loops. Here, the FBs that are mapped onto the devices for *MCU* are summarised in sub-applications. Adding

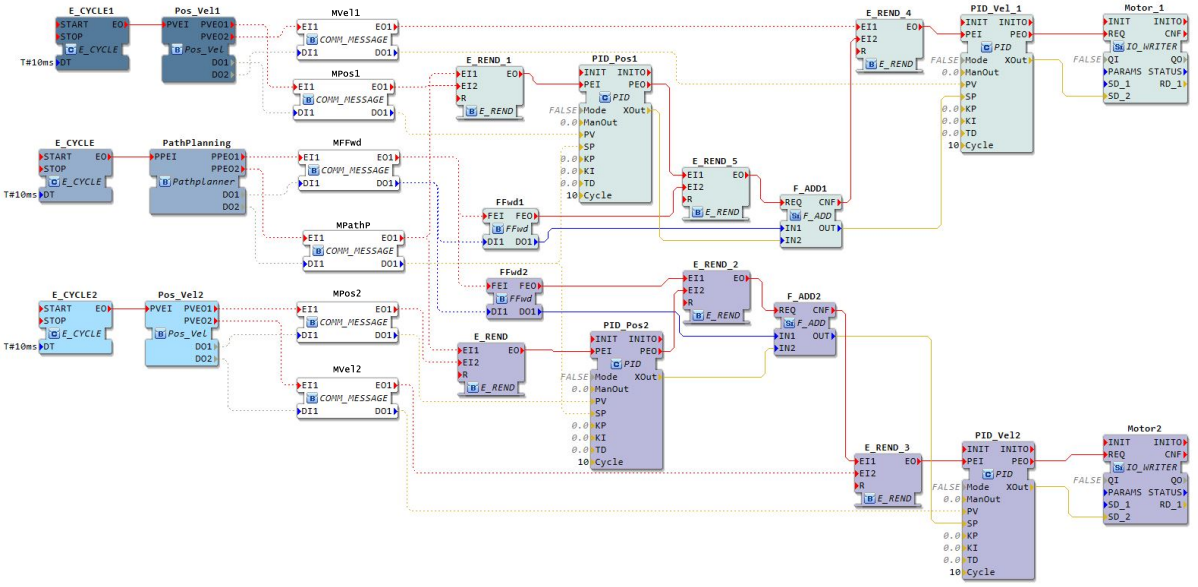


Figure C.6: IEC 61499 control application for two joints with six unmapped messages.

messages for each of the joints' feedback loops results in a total number of eight messages.

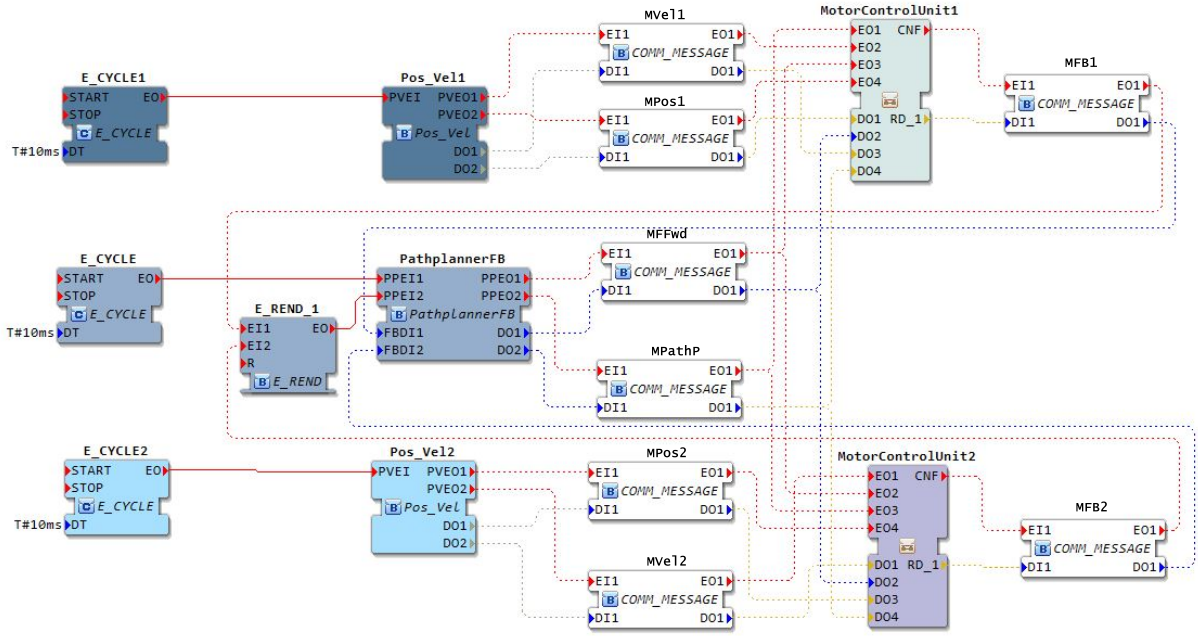


Figure C.7: IEC 61499 joint control application with two joint feedback loops and eight unmapped messages.

Annex D
(informative)
Non-Normative References

1. MACKENZIE, Brendan J.; BRUNS, Friederike; NEBEL, Wolfgang. Model-Based Automation of TSN Configuration for Industrial Distributed Systems. In: *21st IEEE International Conference on Industrial Informatics (INDIN)*. Lemgo, Germany: Institute of Electrical and Electronics Engineers (IEEE), 2023. Available from DOI: 10.1109/INDIN51400.2023.10218085.
2. BRUNS, Friederike; WIESMAYR, Bianca; ZOITL, Alois. Supporting Model-Based Network Specification for Time-Critical Distributed Control Systems in IEC 61499. In: *19th IEEE International Conference on Automation Science and Engineering (CASE)*. Auckland, New Zealand: Institute of Electrical and Electronics Engineers (IEEE), 2023. Available from DOI: 10.1109/CASE56687.2023.10260604.
3. BRUNS, Friederike; MEHLHOP, Sven; WIESMAYR, Bianca; ZOITL, Alois. Enabling Automated Timing Verification: A Unified Approach for Industrial Distributed Control Systems. In: *25th IEEE International Conference on Industrial Technology (ICIT)*. Bristol, UK: Institute of Electrical and Electronics Engineers (IEEE), 2024.
4. BRUNS, Friederike. *Systematic Correct-by-Construction Design for Industrial Real-Time Communication*. Vol. 1. Düren: Shaker Verlag, 2024. Oldenburger Schriften der Verteilten Regelung in vernetzten Systemen. ISBN 9783844096392.
5. ZOITL, Alois; LEWIS, Robert. *Modelling Control Systems Using IEC 61499*. Second. Institute of Electrical and Electronics Engineers (IEEE), 2014. ISBN 978-1-84919-760-1. Available from DOI: 10.1049/PBCE095E.
6. ZOITL, Alois; VYATKIN, Valeriy. Different Perspectives - Face to Face: IEC 61499 Architecture for Distributed Automation: The Glass Half Full View. *IEEE Industrial Electronics Magazine*. 2009, vol. 3, no. 4, pp. 7–23. Available from DOI: 10.1109/MIE.2009.934789.
7. DUBININ, Victor; VYATKIN, Valeriy. On Definition of a Formal Model for IEC 61499 Function Blocks. *EURASIP Journal on Embedded Systems*. 2008, vol. 2008, pp. 1–10. Available from DOI: 10.1155/2008/426713.
8. SEHR, Martin A.; LOHSTROH, Marten; WEBER, Matthew; UGALDE, Ines; WITTE, Martin; NEIDIG, Joerg; HOEME, Stephan; NIKNAMI, Mehrdad; LEE, Edward A. Programmable Logic Controllers in the Context of Industry 4.0. *IEEE Transactions on Industrial Informatics*. 2021, vol. 17, no. 5, pp. 3523–3533. Available from DOI: 10.1109/TII.2020.3007764.
9. WIESMAYR, Bianca; MEHLHOP, Sven; ZOITL, Alois. Close Enough? Criteria for Sufficient Simulations of IEC 61499 Models. In: *19th IEEE International Conference on Automation Science and Engineering (CASE)*. Auckland, New Zealand: Institute of Electrical and Electronics Engineers (IEEE), 2023. Available from DOI: 10.1109/CASE56687.2023.10260555.
10. CENGIC, Goran; AKESSON, Kraut. Definition of the Execution Model Used in the Fuber IEC 61499 Runtime Environment. In: *6th IEEE International Conference on Industrial Informatics (INDIN)*. Daejeon, Korea (South): Institute of Electrical and Electronics Engineers (IEEE), 2008. Available from DOI: 10.1109/INDIN.2008.4618113.
11. CENGIC, Goran; LJUNGKRANTZ, Oscar; AKESSON, Knut. Formal Modeling of Function Block Applications Running in IEC 61499 Execution Runtime. In: *11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Prague, Czech Republic: Institute of Electrical and Electronics Engineers (IEEE), 2006, pp. 1269–1276. Available from DOI: 10.1109/ETFA.2006.355187.
12. SMODIC, Rene; ZOITL, Alois; GRABMAIR, Gunnar; STRASSER, Thomas. A Real-Time Execution Model for IEC 61499 Based Control Applications. *IFAC Proceedings Volumes*. 2006, vol. 39, no. 3, pp. 541–546. Available from DOI: 10.3182/20060517-3-FR-2903.00282. 12th IFAC Symposium on Information Control Problems in Manufacturing.

13. ZOITL, Alois; GRABMAIR, Gunnar; SMODIC, Rene; STRASSER, Thomas. An Execution Environment for Real-Time Constrained Control Software based on IEC 61499. In: *5th IEEE International Conference on Industrial Informatics (INDIN)*. Vienna, Austria: Institute of Electrical and Electronics Engineers (IEEE), 2007. Available from DOI: 10.1109/INDIN.2007.4384932.
14. INSAURRALDE, Carlos C. Fully-Deterministic Execution of IEC-61499 Models for Distributed Avionics Applications. *Aerospace*. 2018, vol. 5, no. 1. Available from DOI: 10.3390/aerospace5010015.
15. DROZDOV, Dmitrii; DUBININ, Victor; PATIL, Sandeep; VYATKIN, Valeriy. A Formal Model of IEC 61499-Based Industrial Automation Architecture Supporting Time-Aware Computations. *IEEE Open Journal of the Industrial Electronics Society*. 2021, vol. 2, pp. 169–183. Available from DOI: 10.1109/OJIES.2021.3056400.
16. SCHWAB, Christian; TANGERMANN, Marcus; LÜDER, Arndt; KALOGERAS, Athanasios; FERRARINI, Luca. Mapping of IEC 61499 Function Blocks to Automation Protocols within the TORERO Approach. In: *2nd IEEE International Conference on Industrial Informatics (INDIN)*. Institute of Electrical and Electronics Engineers (IEEE), 2004. Available from DOI: 10.1109/INDIN.2004.1417319.
17. ATMOJO, Udayanto Dwi; VYATKIN, Valeriy; SALCIC, Zoran. On Achieving Reliable Communication in IEC 61499. In: *IEEE 23rd Int. Conf. Emerging Technologies and Factory Automation (ETFA)*. 2018, pp. 147–154. Available from DOI: 10.1109/ETFA.2018.8502487.
18. PÉREZ, Federico; CALVO, Isidro; LÓPEZ, Fabian; ETXEBERRIA-AGIRIANO, Ismael. Dealing with Communication Channels within IEC61499 Component-Based Systems. *International Journal of Computers*. 2021, vol. 15, pp. 123–129. Available from DOI: 10.46300/9108.2021.15.19.
19. LEDNICKI, Luka; CARLSON, Jan. A Framework for Generation of Inter-Node Communication in Component-Based Distributed Embedded Systems. In: *19th IEEE Emerging Technology and Factory Automation (ETFA)*. Barcelona, Spain: Institute of Electrical and Electronics Engineers (IEEE), 2014. Available from DOI: 10.1109/ETFA.2014.7005222.
20. RIEDL, Matthias; DIEDRICH, Christian; NAUMANN, Felix; ROSE, Nicole. Dependable Distributed Start and Stop. *2006 4th IEEE International Conference on Industrial Informatics*. 2006, pp. 1189–1194. Available also from: https://api.semanticscholar.org/CorpusID:14480480.
21. KIST, Leon; BUSCHMANN, Philippe. Survey on Scheduling Approaches in TSN. In: *Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)*. Munich, Germany, 2022. Available from DOI: 10.2313/NET-2022-11-1_05.