OPC UA based IEC 61499 Device Configuration Interface

Muddasir Shakil
LIT Cyber-Pysical Systems Lab
Johannes Kepler University Linz
Altenberger strasse 69,
4040 Linz, Austria
Email: muddasir.shakil@jku.at

Alois Zoitl
LIT Cyber-Pysical Systems Lab
Johannes Kepler University Linz
Altenberger strasse 69,
4040 Linz, Austria
Email: alois.zoitl@jku.at

***Abstract*—In the modern era of industrial automation, the term Industry 4.0 is defined as the fourth industrial revolution. This is a phenomenon where technologies from various layers of an enterprise are interconnected and form a meshed network of self-regulated, adaptive, re-configurable and self-optimizing devices. These devices vary from Programmable Logic Controller, embedded PCs, edge nodes, smart sensors, and actuators, working as proxies or mediators for a real object in the software domain integrating into Intelligent Enterprise Applications. Heterogeneous configuration interfaces of these devices hinder smooth integration and configuration process. A unified way of interacting with the devices for configuration is well-defined in the IEC 61499 standard. The standard defines the commands, interaction behavior, and interface description for the control devices and engineering tools. There are implementations of the configuration interface in XML and Binary XML, which are widely used for their flexible, extensible, and human-readable nature. Whereas the OPC UA can offer an open configuration interface for the IEC 61499 devices and software tools, with built-in interoperability solutions. This paper introduces the concept of a new configuration interface for the IEC 61499 devices using OPC UA information modeling concepts.**

***Index Terms*—IEC 61499, OPC UA, Configuration interface, Distributed Control System, interoperability.**

I. Introduction

Modern industrial automation systems have evolved into a new generation of self-regulated, self-organized, and interconnected systems. This new industrial revolution is termed as Industry 4.0. In which it is possible to interconnect people, resources, information, and systems to create the Internet of things and services. Industrial Cyber-Physical System (iCPS) is considered as one of the enabling technology for fulfilling the vision of Industry 4.0 [1]. The iCPS is built on the connection between the cyber and the physical world, where cyber part is the digital representation of the real world object, working as a proxy or mediator for a real object in the software domain integrating into Intelligent Enterprise Applications (IEA) [2].

To ensure seamless integration of automation systems into an IEA, OPC UA is proposed as the communication solution for the Reference Architecture Model for Industry 4.0 (RAMI 4.0) [3]. RAMI 4.0 is a framework of standardization and implementation strategies for different domains, hierarchies, and life cycles of an Industry 4.0 component. OPU UA is an IEC standard for interoperability and data exchange also known as IEC 62541. It provides an Object-oriented view of automation systems and access to their functionalities by remote method call services.

On the other hand, the IEC 61499 standard offers a distributed control application modeling language. A Function Block (FB) is the basic building block of the modeling language, encapsulating the control logic and process interface. The control applications are modeled using instances of FBs and their interconnections, known as Function Block Network (FBN). It is an event-driven execution environment, in which data exchange between FBs occur with the triggering of events [4]. The standard provides a generic management model, interface description of the management function block, and command syntax: for the life-cycle management of IEC 61499 applications and interaction with engineering tools [5, pp. 46-50]. Their concrete implementation details are provided by so-called compliance profiles. These profiles describe the interaction behavior, commands structures and exchange formats. A specific compliance profile is developed based on the structure and rules defined in IEC 61499-4. The purpose of the compliance profile is to provide the guidelines for IEC 61499 device, system, and software tool vendors to support the following attributes defined in [6]:

- interoperability among multi-vendor devices.
- portability of software components and system configuration among other software tools.
- configurability of IEC 61499 devices and systems by multi-vendor software tools.

The main aim of this paper is to define the IEC 61499 management services with OPC UA means addressing the device configurability provisions. This is achieved by first

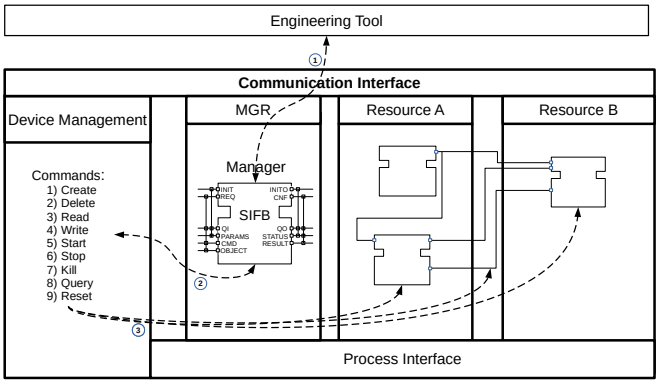

Fig. 1. IEC 61499 device management model (adapted from [7] [8]).

analyzing the management model, management commands, their responses, and the data types used for configuration data exchange in Section II. Furthermore, the overview of the related work is provided in Section III. Moreover, a solution for OPC UA based IEC 61499 configuration interface is presented in Section IV. A proof of concept for validation is presented in Section V and this paper is concluded in Section VI.

II. IEC 61499 Device Management Model

The IEC 61499 configuration approach is based on the management model as illustrated in the Figure 1. The model defines the configuration interface and introduces a device management component for the life-cycle management of resources, control applications, and contained FBs. Furthermore, model adopts service orientation to define management commands. Where configuration requests are passed from an engineering tool to a device via interface exposed by a management resource (MGR). After processing the requested command, the device sends back a standard defined response to the engineering tool. The execution of management commands is handled by the Device Management component shown in the model [4] [7] [8].

The management services are integrated into the run-time by a generic service interface function block (SIFB) known as ”Manager” defined in the Clause 6.3.2 of IEC 61499-1 [5]. Its ”CMD” input parameter specifies the commands for management operation execution. While the ”OBJECT” input defines the structure, and parameters of command. The nine basic commands and status output values are identified in IEC 61499-1 with basic semantics and syntax description. The commands and their semantics are available as shown in Table I.

The syntactic description of the necessary commands for better understanding of the command structure is presented here. These are the basic commands needed for downloading control applications into an IEC 61499 run-time. Therefore, these will be used to demonstrate the proposed solution.

1) Create FB:

```
CMD := CREATE
OBJECT := fb_instance_definition
fb_instance_definition ::=
    fb_instance_reference ':'
    fb_type_name
fb_instance_reference ::=
    [app_hierarchy_name]
    fb_instance_name
app_hierarchy_name :=
    application_name '.'
    {subapp_instance_name '.'}
```

2) Create Connection:

```
CMD := CREATE
OBJECT := connection_definition
connection_definition ::=
    connection_start_point ' '
    connection_end_point
connection_start_point ::=
    fb_instance_reference '.'
    attachment_point
connection_end_point ::=
    fb_instance_reference '.'
    attachment_point
```

3) WRITE:

```
CMD := WRITE
OBJECT := referenced_parameter
referenced_parameter ::=
    [(resource_instance_name |
    fb_instnace_name)'.'] parameter
parameter ::= parameter_name ':='value
```

4) Start

```
CMD := START
OBJECT := fb_instance_reference |
    application_name
```

The standardized responses of the basic commands are described in Table II. An IEC 61499 run-time can generate any of the status output values in context of the requested command.

III. Related Work

The survey of related work in this section is divided in two subsections. The first subsection addresses the work of *The IEC 61499 Compliance Profile for Feasibility Demonstrations*, whereas the second subsection presents a survey on different

TABLE I
IEC 61499 DEFINED BASIC SET OF COMMANDS [5]

Command	Description
Create	Creates a specified object
Delete	Deletes a specified object
START	Starts a specified object
STOP	Stops a specified object
READ	Read a speficied variable
WRITE	Write a specified varible
KILL	Kills an instance immediatly
QUERY	Requests information on a specified object
RESET	Resets a specified object to initial state

OPC UA information modeling approaches for IEC 61499 devices.

A. IEC 61499 Compliance Profile for Feasibility Demonstrations

The IEC 61499 Compliance Profile for Feasibility Demonstrations [9] is the most adopted compliance profile with the concrete definition and implementation details of a configuration interface. As seen in the management model (see Figure 1), each device shall have at least one resource. This resource shall offer application life-cycle management, communication with engineering tools, and implementation of command interfaces. In this compliance profile these types of resources are called RMT_RES. According to [9], each instance of this resource shall contain a special function block called DM_KRNL. It is a composite FB that contains an instance of a service interface FB called DEV_MGR and a TCP/IP client/server communication-based SERVER FB. The DEV_MGR specifies the concrete interface for the MANAGER FB defined in [5, pp. 46-47]. In contrast with the MANAGER FB, the DEV_MGR combines the command and its parameters into one "RQST" input and adds an input parameter which is called "DST", for request destination specification. The result generated by device manager is exposed by "RSP" output parameter and sent back to the configuration application by the SERVER FB [9] [10].

The configuration commands and results are encoded in XML format according to the Request and Response elements described and defined in the Clauses 6.4 and 6.5 of the Compliance Profile [9]. The extensible property of XML format became the basis of extending the basic configuration commands of IEC 61499 to support the reconfiguration tasks. The newly introduced reconfiguration services shall be mapped to XML structure of management commands [4, pp. 79-83].

TABLE II
IEC 61499 DEFINED STATUS OUTPUTS [5]

Value	Status	Description
0	RDY	Command executed succesfully
1	BAD_PARAMS	Requested command has invalid input PARAMS value
2	LOCAL_TERMINATION	Application-initiated termininationn
3	SYSTEM_TERMINATION	System-initiated termination
4	NOT_READY	Manager is not ready to process the command
5	UNSUPPORTED_CMD	Command is not supported
6	UNSUPPORTED_TYPE	Requested object is of unsupported type
7	NO_SUCH_OBJECT	Instance of a requested object is not present
8	INVALID_OBJECT	Syntax description of the command object is invalid
9	INVALID_OPERATION	Requested operation is invalid for specified object
10	INVALID_STATE	Specified object is in invalid state for requested operation
11	OVERFLOW	Previous transactions are still pending

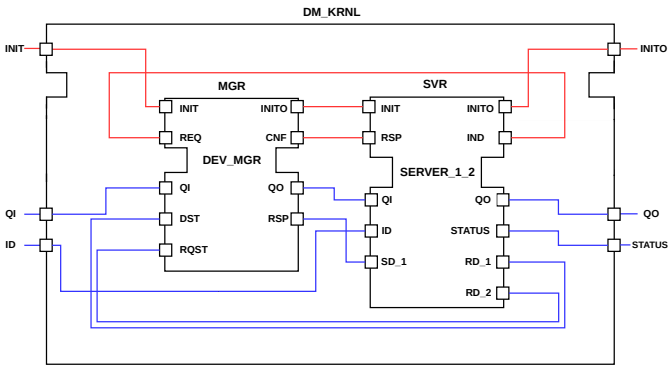

Fig. 2. The IEC 61499 Compliance Profile for Feasibility Demonstrations DM_KRNL Function Block (adapted from [9] [10])

The XML structure of request element consists of Action and ID attributes. ID is a unique identifier assigned to each request to align the responses with their requests and the values of Action attribute specifies the requested command (e.g., CREATE, DELETE, START). The commands require additional data such as FB instance definition or connection definition. The compliance profile used the syntax definition of the command objects from IEC 61499-2 [11].

The work of [10] and [12] compared the textual XML encoding with binary XML encoding. The results suggested that the encapsulation of requests and responses in Binary XML had decreased the parsing effort by the device manager and the size of the request/response message was reduced as well. Both showed that the higher performance can be achieved by switching to Binary XML from String XML format. In the Compliance profile [9], Efficient XML Interchange (EXI) based binary XML encoding has been proposed as FBMGT2 encoding scheme. Binary XML encoding is supported by a new configuration interface called DEV_MGR2, which in contrast with DEV_MGR introduced two new structural data types MGT_REQ and MGT_RSP to model the request and response elements [9].

B. OPC UA Information Modeling Concepts for IEC 61499

OPC UA is being utilized in the IEC 61499 system for various purposes such as data exchange between information systems, service orientation and orchestration, and monitoring and control of the production processes. One approach to create an information model from IEC 61499 FB network was proposed in [13]. They used IEC 61499 Service Interface FB (SIFB) to create, manage, and update OPC UA nodes. The benefit of this approach is that the IEC 61499 control application can actively interact with an OPC UA server. In [14] a new information model for IEC 61499 is proposed, which includes devices, resources, FB, their data variables, events, and connection information. The information model is hosted by a wrapper (OPC UA server), which works as a service mediator between IEC 61499 devices and other software tools. They suggested the concept of using OPC UA information model and node management services to manage

IEC 61499 applications. Another mapping between IEC 61499 and OPC UA information model was presented in [3]. The model was proposed to support the interoperability among various systems and devices. The mapped model can be used to explore the IEC 61499 device hierarchy by any generic OPC UA client. In [15], they also proposed a similar approach modeling IEC 61499 FBs with the OPC UA programs for dynamic discovery and orchestration. Their solution is based on mapping IEC 61499 FBs and applications on to the OPC UA programs. They suggested standard methods in OPC UA programs to deploy the instances of the IEC 61499 FBs. Another mapping suggested in [16], used OPC UA programs to represent IEC 61499 application. Their main objective was to provide a skill-based information model for ease of orchestration and control their execution.

All the approaches proposed different information models to map the IEC 61499 application onto OPC UA. However, there is not a clear guide on how to model IEC 61499 management services in OPC UA address space.

IV. OPC UA For IEC 61499 Management Services

The solution for IEC 61499 configuration interface based on OPC UA standard is presented in this section. It is achieved by presenting the analysis and developed concepts of OPC UA, which are utilized to build the resulting OPC UA based IEC 61499 configuration interface.

A. Analysis of the OPC UA

1) Decoupling Management Services: Generally, the IEC 61499 management commands are based on request and response behavior. The service requester sends a command and expects returned results. Zoitl [4, pp. 225-236] has proposed dedicated SIFB for each management request with a command specific interface. This introduces decoupled implementation for the IEC 61499 application management commands, which can be realized in OPC UA through their implementation with methods and call service set. The OPC UA methods encapsulate the internal functionalities, which is useful for protecting the intellectual properties of the device vendor. Moreover, they support independently deployable management commands, which can be remotely invoked with OPC UA call service set. On other side, with the OPC UA browse service set, engineering tools can dynamically discover the device configuration interfaces. Therefore, in this paper OPC UA methods are adopted to encapsulate and expose the IEC 61499 management requests. The configuration interface inherits OPC UA's built in features like error-handling, and timeout handling.

2) Device Manager: By analyzing IEC 61499 device management model in Figure 1, it was realized that all management commands are contained and organized by the Device Management component. It is the point of interaction between the communication interface and internal logic of commands. Therefore, "Device Manager" node of base object type is introduced in the OPC UA information model. The purpose of this node is to organize and provide a dedicated browse path to the command method nodes. All the command methods are organized under this object node.

3) Application Hierarchy: In the OBJECT element of the create and write commands the `fb_instance_reference` and `referenced_parameter` data types are used to specify the hierarchical structure of the control application. They usually start with resource instance name followed by application, sub-application instance names, and finally fb instance name. In case of `referenced_parameter`, it ends with the parameter name of the specified object. OPC UA's dynamic array of string data type can be used to represent the `fb_instance_reference` and `referenced_parameter`. Dynamic arrays can be utilized to handle the varying IEC 61499 application hierarchy. The value on each index of the array refers to an element of the application hierarchy. The last index of the array points to the fb instance name for create commands and parameter name in case of `referenced_parameter`.

4) Destiniation Parameter: The control applications are created, managed, and executed inside a resource; therefore, resource instance name is specified for each command using the destination parameter. The destination of the management request is specified by "DST" input variable (Figure 2) in IEC 61499 Compliance Profile for Feasibility Demonstrations [9]. If the value is an empty string, meaning that device is the target or if it contains a resource identifier than the specified resource is the request's target [17, p. 177]. A similar approach is adapted by introducing a destination input argument for device management OPC UA methods. OPC UA specific string data type known as "UA_STRING" is assigned to the destination parameter.

5) Output Status: The returned output STATUS codes define the semantic of the results. These status codes are mapped to an enumeration list. The enumeration values point to their respective IEC 61499 specific STATUS output codes. The values, status codes, and their semantics are shown in Table II. Enumerated values are exchanged like numeric values over the connection and therefore, impose less traffic as compared to string data types. Another advantage of enumerated values that they help programmers to write logical code on values.

6) Change State Commands: IEC 61499 currently defines four state changing commands: start, stop, kill, and reset. At the interface level all state related commands are homogeneous. Grouping them with a dedicated OPC UA method is the next logical step. The newly introduced method is called "mgm_changeState". State commands are mapped to an enumeration list known as "enumStateCommand",

TABLE III
Enumeration list for the state related commands

Value	State	Description
0	Start	Starts an object
1	Stop	Stops an object
2	Kill	Kills an object
3	Reset	Resets an object

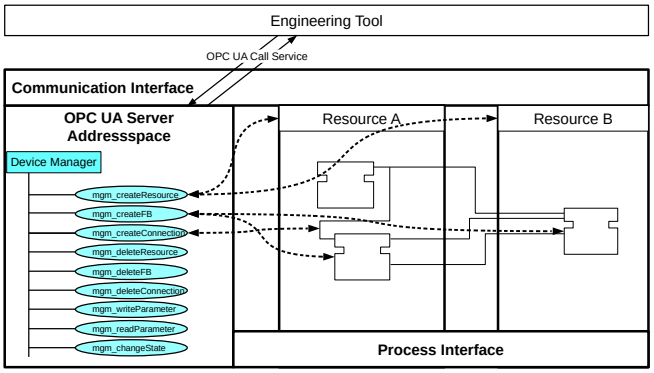

Fig. 3. Device management model based on proposed solution

which are passed as input argument for "'mgm_changeState" method. This approach reduces the number of methods to be implemented for IEC 61499 configuration interface and simplifies the extension of state related commands because less effort will be required to extend the list at interface level without adding new methods. The base state values are presented in the Table III.

B. Resulting IEC 61499 Configuration Interface

The new IEC 61499 device model based on OPC UA configuration interface is shown in Figure 3. The management resource is now replaced with the OPC UA server, in which IEC 61499 management commands are modeled using OPC UA method nodes.

The Table IV, shows the OPC UA methods implementing dedicated IEC 61499 management commands. These methods are developed based on the concepts discussed in Section IV-A. Input parameters with "_Name" suffix are simple identifiers of string data type. They are used to specify type of a resource or FB, instance name of a resource, and destination of the targeted FB. If resource is the request's target than, Destination_Name must be an empty string. In IEC 61499 the connection between two FB instances in different resources is not allowed. This condition checking is guaranteed by introducing destination parameter in connection request methods. The application and resource hierarchy of a FB as discussed in Section IV-A is defined by input arguments with "_Reference" suffix. The hierarchy is upgraded by dynamic arrays of OPC UA string data type. The STATUS output codes are mapped to the enumeration list as discussed in Section IV-A and assigned as output arguments to the methods. OPC UA String data type is assigned to the Value parameter and a proper conversion must be provided in the device from string.

V. Prototype Implementation

A prototype was developed with 4diac FORTE [18] for the validation of the concept. 4diac FORTE is an open source IEC 61499 run-time environment. An OPC UA server using Open62541 [19] stack was implemented in the 4diac FORTE. The IEC 61499 configuration interface was implemented using methods in the information model of this server. The run-time has already integrated Open62541's [19] stack through communication layer and handler. A deployment client was also built in the 4diac IDE [18] using Eclipse-Milo [20] open source OPC UA Java based stack. The deployment client handles the exchange of the command requests and responses between 4diac IDE and the OPC UA server hosted by 4diac FORTE. For demonstration of the developed solution, a generic OPC UA client called UaExpert provided by Unified Automation [21] is used. In the demonstration, UaExpert represents any IEC 61499 configuration tool that can integrate OPC UA as a deployment client. It shows the broader applicability and interoperability of the developed solution.

Figure 4 shows the basic deployment process using OPC UA methods. It shows the interfaces of the developed OPC UA methods to create resources, FBs and connections, and changing the state of a device, resource, and FB. In Figure 4, the configuration command sequence contains creating a resource instance in the device (4a), creating FB instances by calling "mgm_createFB" method (4b), creating connections between FBs (4c), and triggering execution of FBs by starting the Resource (4d). Although a simple IEC 61499 application was chosen but it allowed testing the operation of the prototype implementation and validation of the developed concept of OPC UA based IEC 61499 device configuration interface.

VI. Conclusion and Outlook

This paper describes an approach for a service-oriented IEC 61499 device configuration interface using OPC UA. It enabled to develop an interoperable and a generic communication solution for IEC 61499 architecture. Moreover, the IEC 61499 management commands can be modeled with OPC UA methods. These methods allowed to implement device management commands in a decoupled fashion. With

TABLE IV
OPC UA Methods for IEC 61499 Management Request

Method BrowseName	Input Argument	Output Argument
mgm_createResource	Instance_Name Type_Name	STATUS output code
mgm_createFB	Destination_Name Instance_Reference Type_Name	
mgm_createConnection	Destination_Name Source_Reference Destination_Reference	
mgm_deleteResource	Instance_Name	
mgm_deleteFB	Destination_Name Instance_Reference	
mgm_deleteConnection	Destination_Name Source_Reference Destination_Reference	
mgm_writeParameter	Destination_Name Parameter_Reference Value	
mgm_changeState	Destination_Name Instance_Reference State	
mgm_readParameter	Parameter_reference	STATUS output code Value

(a) Creating a Resource

(b) Creating a FB

(c) Creating a connection

(d) Starting the Resource

Fig. 4. UaExpert views for developed OPC UA methods

the use of OPC UA methods, the configuration interface becomes browsable, which can be discovered dynamically by other IEC 61499 configuration tools. In the end, the interoperability of the solution is demonstrated by a prototype implemented in an IEC 61499 run-time, and interacted with it through a generic OPC UA client.

The future work is related to investigating the OPC UA programs in regard with the IEC 61499 device configuration and integrating OPC UA methods to control the execution of the deployment process. Furthermore, the performance of the developed solution and other approaches for IEC 61499 device configuration will be analyzed and compared. Moreover, the security of the configuration interface will be investigated, and suitable solutions will be presented.

REFERENCES

[1] K. Henning, "Recommendations for implementing the strategic initiative industrie 4.0," 2013.

[2] D. Repta, A. M. Stanescu, M. A. Moisescu, I. S. Sacala, and M. Benea, "A cyber-physical systems approach to develop a generic enterprise architecture," in *2014 International Conference on Engineering, Technology and Innovation (ICE)*, June 2014, pp. 1–6.

[3] W. Dai, Y. Song, Z. Zhang, P. Wang, C. Pang, and V. Vyatkin, "Modelling industrial cyber-physical systems using iec 61499 and opc ua," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, July 2018, pp. 772–777.

[4] A. Zoitl, *Real-Time Execution for IEC 61499*. ISA, 2008.

[5] *IEC 61499-1, Function Blocks - Part 1: Architecture*, International Electrotechnical Commission Std., 2011.

[6] *IEC 61499-4, Function Blocks - Part 4: Rules for Compliance Profiles*, Geneva: International Electrotechnical Commision Std., 2012.

[7] T. I. Strasser, M. N. Rooker, G. Ebenhofer, and A. Zoitl, "Standardized dynamic reconfiguration of control applications in industrial systems," 2014.

[8] F. Andrén, T. Strasser, A. Zoitl, and I. Hegny, "A reconfigurable communication gateway for distributed embedded control systems," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 3720–3726.

[9] "IEC 61499 Compliance Profile for Feasibility Demonstrations," Jan 2019, [Online; accessed 18. Jun. 2019]. [Online]. Available: https://www.holobloc.com/doc/ita/index.htm

[10] A. Zoitl, I. Hegny, and A. Schimmel, "Utilizing binary xml representations for improving the performance of the iec 61499 configuration interface," in *2009 7th IEEE International Conference on Industrial Informatics*, June 2009, pp. 66–71.

[11] *IEC 61499-1, Function Blocks - Part 2: Software requirements*, International Electrotechnical Commission Std., 2011.

[12] F. Noack, "Evaluation of binary xml for configuring industrial control systems," Bachelor's Thesis, Technische Universität München, 02 2016.

[13] T. Terzimehic, M. Wenger, A. Zoitl, A. Bayha, K. Becker, T. Müller, and H. Schauerte, "Towards an industry 4.0 compliant control software architecture using iec 61499 opc ua," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2017, pp. 1–4.

[14] I. Seilonen, V. Vyatkin, and U. D. Atmojo, "Opc ua information model and a wrapper for iec 61499 runtimes," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 1008–1013.

[15] M. Kaspar, J. Bock, Y. Kogan, P. Venet, M. Weser, and U. E. Zimmermann, "Tool and technology independent function interfaces by using a generic opc ua representation," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 1183–1186.

[16] K. Dorofeev and A. Zoitl, "Skill-based engineering approach using opc ua programs," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. IEEE, 2018, pp. 1098–1103.

[17] A. Zoitl and R. Lewis, *Modelling control systems using IEC 61499*. IET, 2014, vol. 95.

[18] "Eclipse 4diac - The Open Source Environment for Distributed Industrial Automation and Control Systems," Sep 2019, [Online; accessed 11. Sep. 2019]. [Online]. Available: https://www.eclipse.org/4diac

[19] "open62541: an open source implementation of OPC UA," Sep 2019, [Online; accessed 11. Sep. 2019]. [Online]. Available: https://open62541.org

[20] eclipse, "milo," Aug 2019, [Online; accessed 11. Sep. 2019]. [Online]. Available: https://github.com/eclipse/milo

[21] "UaExpert "UA Reference Client"," Feb 2019, [Online; accessed 16. Sep. 2019]. [Online]. Available: https://www.unified-automation.com/products/development-tools.html