Modelling Industrial Cyber-Physical Systems using IEC 61499 and OPC UA

Wenbin Dai, *IEEE Senior Member*, Shanghai Jiao Tong University, China, w.dai@ieee.org
Yineng Song, Zhijie Zhang, Shanghai Jiao Tong University, China 15821393690@139.com, 825550026@qq.com
Peng Wang, Shenyang Institute of Automation, Chinese Academy of Science, wangpeng@sia.cn
Cheng Pang, Jiangmen Goobotics Research Institute, Zhuxi Wisdom Valley, Jiangmen, China, cheng.pang.phd@ieee.org
Valeriy Vyatkin, *IEEE Senior Member*, Luleå University of Technology, Sweden and Aalto University, Finland, vyatkin@ieee.org

***Abstract* – Industrial Cyber-Physical Systems (iCPS) are considered as the enabling technology for achieve Industry 4.0. One main characteristic of the iCPS is the information transparency to allow interoperability among various devices and systems. The OPC UA provides a common information model for connecting Industry 4.0 components. On the other hand, the IEC 61499 is commonly used as an executable modeling language for iCPS. The IEC 61499 function block network provides an abstract view of the system configuration. By combining IEC 61499 and OPC UA, a visual executable model for iCPS is completed. In this paper, the mapping between two standards are provided and a case study of the proposed mapping is given.**

***Index Terms* —** *Industrial Cyber-Physical Systems; Executable Modelling Language; IEC 61499 Function Blocks; OPC-UA; Interoperability.*

I. INTRODUCTION

The 4th Industry revolution or "Industrie 4.0" [1] aims to lift the informatics level of legacy industrial automation to the next level. The industrial cyber-physical systems (iCPS) [2] or cyber-physical production systems (CPPS) is considered as the enabling technology for fulfilling Industry 4.0. The iCPS is targeting for bridging the physical plant with the digital control systems as closed-loop by feedback information [3]. The Industry 4.0 working group published RAMI 4.0 reference architecture that mainly focus on technologies and standards integration [4] in the ISA-95 architecture [5].

Peer-to-peer data exchange must be enabled as information transparency is the fundamental requirement of the iCPS. To ensure seamless integration between all nodes, the OPC UA (also known as the IEC 62541 standard) [6] is proposed as the only communication solution for the RAMI 4.0. The OPC UA standard from the OPC foundation is an extension of the famous OPC DA standard with a set of new features such as publish/subscribe, discovery and time-sensitive network (TSN) [7]. The OPC UA provides an object-oriented view of automation devices and methods to access internal functions inside objects.

On the other hand, the IEC 61499 standard is referred as the executable modelling language for distributed automation systems [8]. System configurations are presented in function block networks that provides encapsulation of control logics, communication handlers, as well as human machine interfaces (HMI). The data exchange between function blocks are defined by event and data connections in an abstract way. These abstract communication models can be fulfilled using any protocol that are not limited by the standard itself. In this case, the OPC UA is the most suitable solution to enhance existing abstract communication models with semantics and a standard way to access function blocks from external environment. In this paper, the modelling techniques using IEC 61499 and OPC UA is proposed to allow information transparency in the iCPS.

The rest of the paper is organized as follows: In Section II, related works of adopting the OPC UA in industrial automation domains are investigated. In Section III, the information model of the OPC UA and the IEC 61499 function blocks are compared and mapping rules are provided. The heterogeneous modelling method for the iCPS is described in this Section based on IEC 61499 and OPC UA. Following that, the implementation of integration the OPC UA with the IEC 61499 on the runtime level is illustrated. The proposed modelling language is tested on a sortation system in Section IV. Finally, conclusions and future works are provided.

II. RELATED WORKS

OPC UA has already shown its impact in industrial automation. On the control level, Melik-Merkumians et. al. [9] proposed a SOA and OPC-UA based middleware for connecting information between the control level and the process control level, the operations level and the enterprise level. The OPC UA compliance profiles for IEC 61131 and IEC 61499 are illustrated. The results prove that merging SOA with OPC-UA is a feasible solution for industrial control middleware.

OPC UA provides object-oriented programming concept for the generic information model in iCPS [10]. Kozar et. al. extends the IEC 61499 application with OPC UA communication protocol [11]. The open62541 OPC UA protocol stack is integrated with the 4DIAC framework with write and read service interface function blocks (SIFB). Also, the OPC UA publish and subscribe SIFBs are implemented to enhance original pub/sub in IEC 61499 with OPC UA manner.

On the management level, a semantic integration of OPC UA information model with manufacturing execution systems (MES) is proposed by Schleipen et. al. [12]. A graphical tool for modelling OPC UA server address space is developed. In addition, semantics predefined in the CAEX format are also integrated with the OPC UA information model.

Henben et. al. also experimented interoperability between OPC UA and AutomationML [13]. The AutomationML

information model is mapped to the OPC UA information model by exchanging the data formats. The OPC UA server address spaces already created from AutomationML information model could be reused that increases efficiency of developing new OPC UA information models. Additionally, security issues are discussed in [14] on OPC UA and AutomationML models. A role-based security concept is proposed and implemented in a software tool.

Pauker et. al. provides a systematic approach for designing OPC UA information model [15]. The I4.0 components defined in the RAMI 4.0 are proposed to use SOA based communication. The OPC UA model is used to be represent static and dynamic behavior of system semantics. The overall design process of OPA UA information model for manufacturing processes is simplified by using this approach.

Hastbacka et. al. adopted the OPC UA for device status condition monitoring in operation and maintenance [16]. The device status information from sensors and field devices are gathered by the OPC UA and can be directly accessed from enterprise level applications and services.

The OPC UA is integrated in the smart grid as well. Rohjans et. al. combined the OPC UA model with the IEC 61970/61968 common information model (CIM) to up lift the semantic level between Energy Management Systems (EMS), Distribution Management Systems (DMS) and Enterprise Resource Planning (ERP) Systems [17]. Similar work is also presented by Lehnhoff et. al. [18] in the smart grid research. The OPC UA is mapped with the IEC 61850 protocol for future-proof communication models between substations.

Overall, OPC UA is well proven in many industrial applications as the key to enabling information transparency. The PLCOpen published the official integration guide of IEC 61131-3 with the OPC UA model [19]. However, there is not a clear guide on how to map and browse the IEC 61499 information model with OPC UA yet.

III. Mapping IEC 61499 and OPC UA Information Model

The OPC UA provides a base information model for interoperability between various devices and systems in iCPS. On the other hand, the IEC61499 standard also specifies information model for distributed automation systems. There is yet a clear guideline on mapping between the OPC UA and the IEC 61499 information model. In this section, both models will be described and the mapping rules are provided.

A. IEC 61499 Information Model

The IEC 61499 standard is considered as an executable modeling language for iCPS [20]. Control algorithms, human machine interface (HMI) and communication functions are encapsulated in a programming organization unit (POU) – function block (FB). A function block provides an interface that provides inputs and outputs connected to other function blocks. Differ from the IEC 61131-3 FBs [21], the IEC 61499 FBs [22] are triggered by events which means control flows are separated from data flows. This provides great flexibility and interoperability for inter-device and system-to-system cooperation.

Several software models are defined in the IEC 61499 standard. As shown in Fig. 1, the system configuration model contains one or more applications that each application is represented by a function block network and could be deployed to one or more devices. In each device model, several resources could be setup to execute multiple applications. Resources and are communicating with each other via process interfaces.

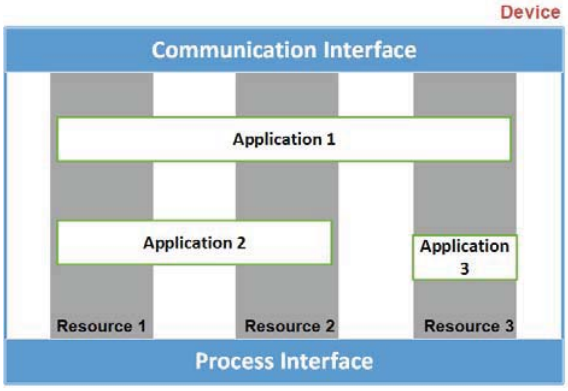

Fig. 1: The IEC 61499 Device Model [22]

In each resource, a function block network model is used to provide control, HMI or communication functions. Instances in a function block network are connected by event and data connections. As mentioned earlier in this section, function blocks are triggered by events. When an output event is raised, data outputs associated with this particular event will be updated to downstream FBs via data connections. Event and data inputs and outputs are defined in the FB interface.

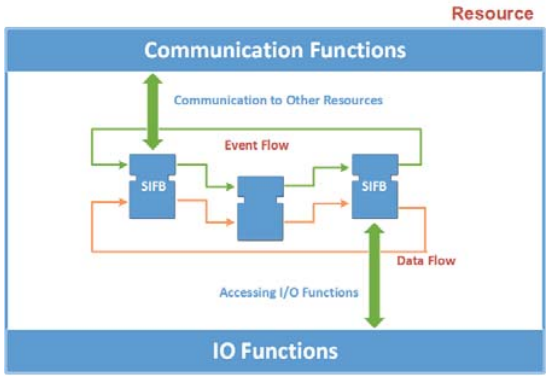

Fig. 2: The IEC 61499 Resource Model [22]

Function blocks are divided into two categories: atomic function block type and composite function block type. For atomic FBs, algorithms written in IE 61131-3 programming languages, high level programming languages or HMI configurations are encapsulated. For composite FBs, another function block network is encapsulated that consists of both atomic and composite FBs.

B OPC UA Information Model

Now, the OPC UP information model will be presented briefly. As illustrated in Fig. 4, the base OPC UA information

model is designed based on object-oriented programming concepts and each OPC UA server usually has multiple nodes defined by *NodeClasses* represents *Objects*, *Variables* and *Methods* [23]. The *Variable* is used to represent a value with a specified data type, for example, a proximity sensor that indicates object presents or a speed odometer. The *Methods* refer to a function all with input arguments and return results, for instance, a method could be start and stop a conveyor. The *Objects* are used as entities for encapsulating *Variables* and *Methods* as well as handling events generated internally.

Between nodes, several reference types could be configured including *HasComponent*, *HasProperty*, *HasTypeDefinition*, *HasSubtype, HasEventSource* and etc. A reference could be hierarchical or non-hierarchical. For non-hierarchical references, several properties could be identified such as *HasModelParent, GeneratesEvents, HasEncoding, HasModellingRule, HasDescription, and HasTypeDefinition*. For hierarchical references, child/parent nodes could be specified by *HasChild*, *Organizes* and *HasEventSource*.

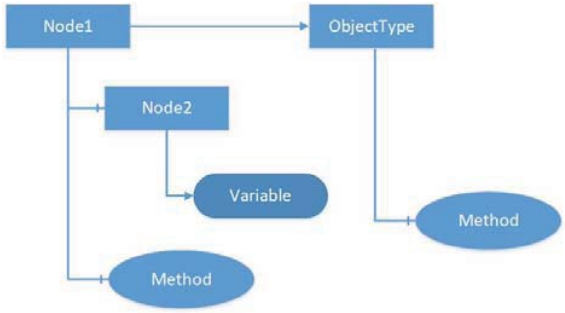

Fig. 4: Base Information Model Reference for OPC UA

Objects are divided into simple and complex object types. Similar to the inheritance in the OOP concept, a complex object can be extended from a simple object types that all variables and methods are accessible from the new complex object type.

C Mapping IEC 61499 with OPC UA Information Model

To achieve interoperability between IEC 61499-based devices with other devices, the information models between IEC 61499 and OPC UA must be mapped. As shown in Fig. 3, the top level entity of the OPC UA information model is mapped to an IEC 61499 resource instance, namely *FB_DevInst.* A *FB_DevInst* can contain one or more IEC 61499 resources - *FB_ResInst.* A *FB_ResInst* may be used as a holder for one or more independent applications – *FB_Network*. *FB_Networks* are built from a number of inter-connected function block instances and Sub-application instances, namely *FB_Inst*, written in any IEC 61131-3 programming languages or even high-level programming languages. All *FB_DevInsts*, *FB_ResInsts*, *FB_Networks* and *FB_Insts* are modelled as OPC UA *Objects*. The *HasComponent* reference is used for describe the hierarchy structure, where:

FB_DevInst HasComponent FB_ResInst
FB_ResInst HasComponent FB_Network
FB_Network HasComponent FB_Inst

In the OPC UA, two nodes of *Object* are connected by a *Reference*. The semantics between various nodes are defined by the *References*. A *Reference* can be directly accessed in the OPC UA address space as nodes. A *Reference* must be declared as a *ReferenceType* before it can be applied. A *ReferenceType* could be a non-hierarchy (cannot be extended) or hierarchy (scalable).

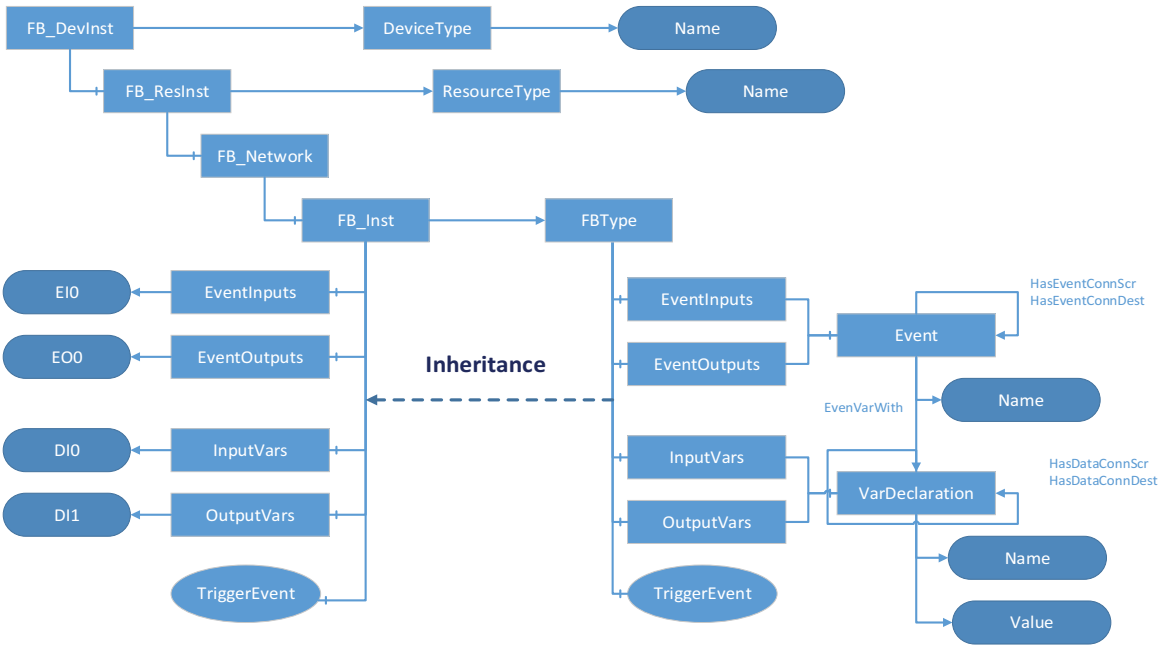

Fig. 3: Mapping IE 61499 Applications to OPC UA information model

In an IEC 61499 application, function block instances are linked by two categories of connections: event connections and data connections. Event connections are mainly used to model control flow through function block networks. Values of variables are exchanging between FBs via data connections. In the case of the OPC UA, both event and data connection are modelled as *ReferenceTypes*, namely *HasEventConnScr* and *HasDataConnScr* from source sides of connections. As both *ReferenceTypes* are not abstract and non-symmetric, the inverse names are set as *HasEventConnDest* and *HasDataConnDest* on destination sides of connections.

Each OPC UA object must also be referred to an *ObjectType*. In this case, each instance of IEC 61499 device, resource and function block instances is assigned with an *ObjectType,* namely *DeviceType, ResourceType and FBType*. For a *DeviceType*, a string attribute *Name* must be specified and one or more resources with Object Type of *ResourceType* could be organized. Similar to the *DeviceType*, a *ResourceType* also attaches a string attribute of *Name* as well as exact one *FB_Network* that contains one or more FB instances connected with event connections, data connections and adapter connections. Those FB instances are referred as FB Types that represents a BFB, SIFB, CFB or SubApp FB type.

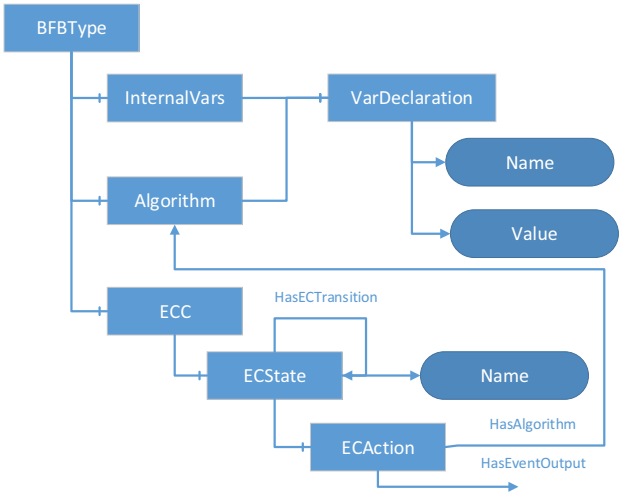

Fig. 5: Basic FB Type Information Model for OPC UA

As shown in Fig. 5, OPC UA base FB Type objects have a component of the interface where input/output event and data variables are defined. An *FBType* object organizes zero or one *EventInputs, EventOutputs, InputVars, OutputVars, Sockets* and *Plugs* object. For *EventInput* and *EventOutput*, zero or more *Events* are declared that *Event* has data attributes of *Name* and *Type*. In addition, Each *Event* object also attaches a method *Trigger* that allows triggering events externally. For *InputVars and OutputVars,* zero or more *VarDeclarations* are organized that each variable declaration has attributes of name, type, array size and initial value. Three methods are defined for each *VarDeclaration – Read* value from variable, *Write* value to variable and *Override* value for this variable. *VarDeclarations* could be linked with *Events* by a reference type of *EventVarWith*. When this event trigger method is invoked, associated variables must be also read as defined in the IEC 61499 standard. *Plugs* and *Sockets* refer to an *AdapterDeclaration* object type which consists two attributes of *Name* and *Type* and bunch of *Parameters* as objects.

Four different FB types *BFBType*, *SIFBType*, *CFBType* and *SubAppType* are inherited from the base FB type. As shown in Fig. 5, for the BFB object type, three sub objects are organized: internal variables (*InternalVars*), execution control chart (*ECC*) and code (*Algorithm*). For internal variables, the same object *VarDeclaration* is utilized. For the ECC, there are some EC states that has two data attributes of name and current status as well as organizing zero or more *ECActions*. The EC actions has two customized references that link to zero or more *Algorithm* and output *Events*. EC actions are joined together by EC transitions. EC transitions are modelled as a data attribute of *Condition* with two references to source and destination EC states. An *Algorithm* object could have some temporary variables as the *VarDeclaration* node type.

For a SIFB type, one or more service sequences could be declared which has one or more service transitions as illustrated in Fig. 6. For each service transition, one Input primitive and zero or more Output primitive could be defined. In each primitive, three data attributes are attached including *Interface, Event* and *Parameters*.

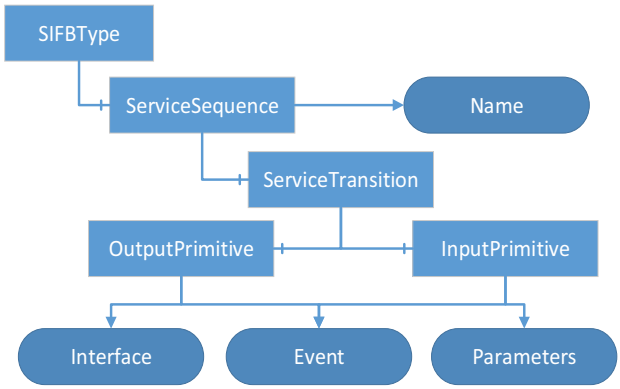

Fig. 6: Service Interface FB Type Information Model for OPC UA

Finally, for CFB and SubApp types, a FB network node is defined that is identical to the *FBNetwork* defined in the resource model.

IV. Implementation and Case Study

The proposed model mapping between IEC 61499 and OPC UA is implemented in the Function Block Service Runtime (FBSRT) [24] with integration of the Open62541 OPC UA protocol stack [25]. An OPC UA server is built for each FBSRT runtime as an IEC 61499 device. The OPC UA information model of IEC 61499 applications are created upon system deployment. When a FB instance is modified or deleted, the corresponding OPC UA model will also be updated or deleted simultaneously.

As shown in Fig. 7, a case study of the parcel sortation system is used to verify the OPC UA information model for IEC 61499 FBs. The parcel sortation system starts with a induction conveyor, followed by a turntable that rotates parcel for 90 degrees. Finally, the parcel will be sorted into one of the

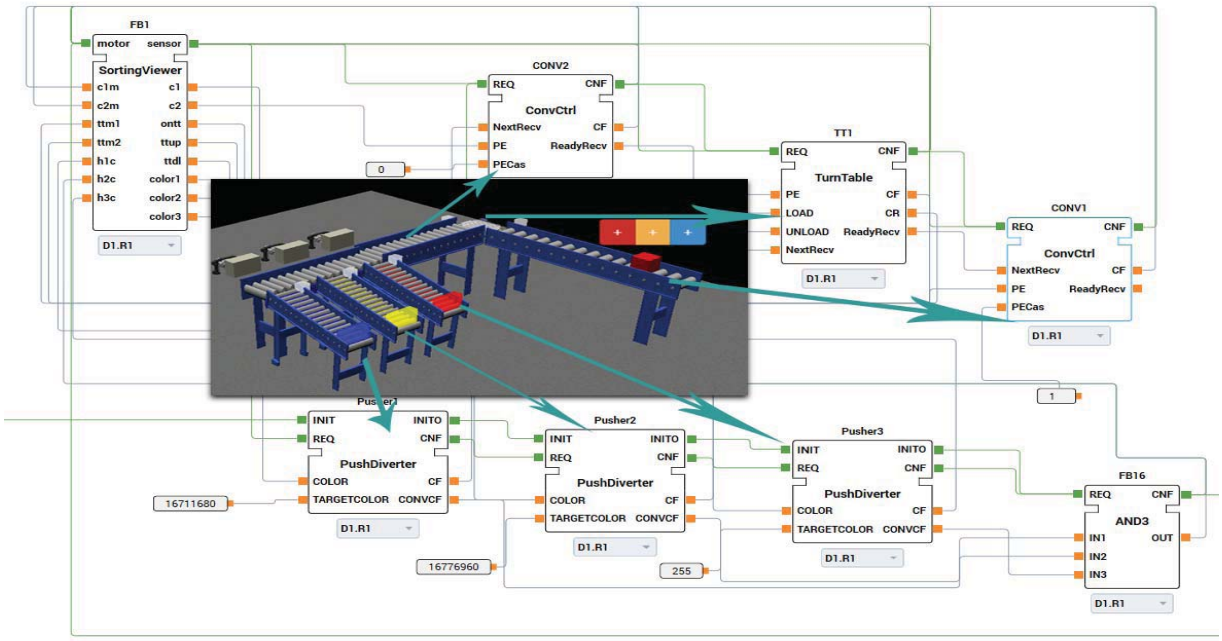

Fig. 7: Case Study of A Parcel Sortation System and FB Implementation

three chutes holding blue, red and yellow parcels. The object-oriented programming is also applied in the function block implementation of the parcel sortation system. Each conveyor (*CONV1* and *CONV2*), turntable (*TT1*) and sorter (*Pusher1*, *Pusher2* and *Pusher3*) are modelled as an individual FB in the network. Those object FBs are connected according to their geometric order *(CONV1 -> TT1 -> CONV2 -> Pusher1 -> Pusher2 -> Pusher3)*.

An OPC UA client is connected to the FBSRT to explore the information model of the sortation system. As illustrated in Fig. 7, the FBSRT OPC UA server has one resource deployed namely *D1.R1*. When exploring the IEC 61499 resource *D1.R1*, an IEC 61499 application *a1* is configured. Inside this function block network, several FB instances are listed including conveyor control FBs *CONV1* and *CONV2*, turntable control FB *TT1*, diverter control FBs *pusher1, pusher2* and *pusher3*. Browsing the *CONV1* FB, three data inputs (*NextRecv*, *PE* and *PECas*) and two data outputs (*CF*, *ReadyRecv*) are defined in the *ConvCtrl* FB object type.

Each conveyor control block has an event *REQ* with two inputs of photo-eye sensor *PE* to detect object presents on the belt and a signal *NextRecv* from downstream conveyor that can receive objects from upstream. It also has event output *CNF* with two Boolean outputs *CF* for controlling motors run/stop and *ReadyRecv* for indicating can receive objects from upstream conveyor. In the ECC of the conveyor control FB, two states *RUN* and *STOP* are indicating the physical status of the conveyor.

As seen in Fig. 7, similar to the conveyor control block, the turntable control block has additional two inputs *LOAD* and *UNLOAD* to indicate the rotation position and an output *CR* for running in the reverse direction. However, the ECC of the turntable contains idle, load, turn forward, unload and turn backward. The push diverter FB is relatively simple with a sensor of detecting objects and two outputs that is controlling push action and the underneath conveyor running.

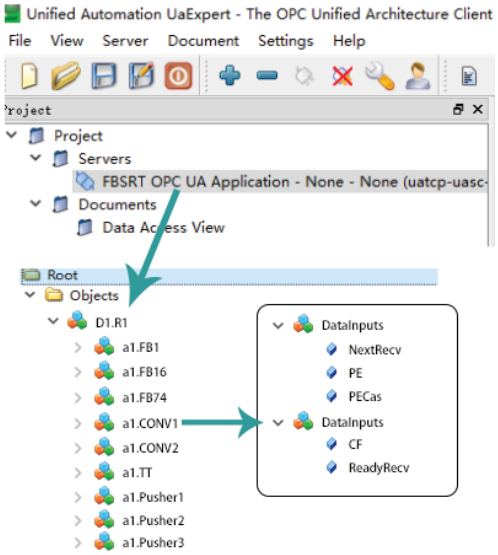

Fig. 8: OPC UA Information Model for the IEC 61499 based parcel sortation system.

As shown in the Fig. 8, an OPC UA client is used to verify the results of the case study. The IEC 61499 runtime is discovered by the OPC UA client and by browsing through the objects defined in the IEC 61499 runtime, all FB instances in the Fig. 7 are listed. For each FB instance, all data inputs and

outputs can be read and written by standard OPC UA clients. In addition, data inputs and outputs can be overridden by OPC UA clients with fixed values.

Overall, from the case study, the hierarchy of the IEC 61499 model could be explored through the standard OPC UA object view. Data variables of inputs, outputs and internals can be directly accessed using OPC UA clients.

V. Conclusions And Future Work

The OPC UA provides a common information model for exchanging data between various industrial automation devices and subsystems. The OPC UA is adopted for many industrial standards like IEC 61131-3. In this paper, the OPC UA information model for the IEC 61499 standard is mapped. The hierarchy of the IEC 61499 system models can be explored through OPC UA clients. Variables and state machine status could be retrieved from IEC 61499 runtime level. In addition, events and data variables could also be triggered and overridden manually through OPC UA clients. The results demonstrate the IEC 61499 and OPC UA information models are mapped as designed and can be integrated with other OPC UA information models, for example IEC 61131-3 for interoperability.

Continuously from this work, the management model and the data model of the IEC 61499 will be mapped to the OPC UA version. Vice versa, the OPC UA event models will be applied to the IEC 61499 events. Finally, the interoperability tests between IEC 61499 and IEC 61131-3 devices as well as the SCADA/HMI systems will be performed.

Acknowledgements

This research work is sponsored by the National Key R&D Program of China (2017YFA0700602) and the National Science Foundation, China (Project No. 61503247).

References

[1] H. Kagermann, J. Helbig, A. Hellinger and W. Wahlster, “Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry”, *Final Report of the Industrie 4.0 Working Group*, Forschungsunion, 2013

[2] A. Colombo, S. Karnouskos, and T. Bangemann. "Towards the next generation of industrial cyber-physical systems." *Industrial cloud-based cyber-physical systems*. Springer International Publishing, 2014. 1-22.

[3] P. Nuzzo, A. Sangiovanni-Vincentelli, and R. Murray, "Methodology and Tools for Next Generation Cyber-Physical Systems: The iCyPhy Approach," in *25th Annual INCOSE International Symposium (IS 2015)*, Seattle, WA, US, 2015.

[4] M. Hankel and B. Rexroth, The reference architectural model industrie 4.0 (rami 4.0). *ZVEI*, April, 2015.

[5] B. Scholten, “The road to integration: A guide to applying the ISA-95 standard in manufacturing”, *ISA*, 2007.

[6] F. Palm, S. Grüner, J. Pfrommer, M. Graube, and L. Urbas, “Open source as enabler for OPC UA in industrial automation”, *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1 – 6, 2015.

[7] S. Kehrer, O. Kleineberg, and D. Heffernan, “A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN)”, *IEEE International Conference on Emerging Technology and Factory Automation (ETFA)*, pp. 1-8, 2014.

[8] A. Zoitl, and H. Prähofer, “Guidelines and patterns for building hierarchical automation solutions in the IEC 61499 modeling language”, *IEEE Transactions on Industrial Informatics*, Vol. 9, No. 4, pp.2387-2396, 2013.

[9] M. Melik-Merkumians, T. Baier, M. Steinegger, W. Lepushitz, I. Hegny and A. Zoitl, “Towards OPC UA as portable SOA Middleware between Control Software and External Added Value Applications”, *IEEE 17th Conference on In Emerging Technologies & Factory Automation (ETFA)*, pp 1 – 8, 2012.

[10] A. Maka, R. Cupek, J. Rosner, “OPC UA object oriented model for public transportation system”, *IEEE Fifth UKSim European Symposium on Computer Modeling and Simulation (EMS)*, pp. 311-316, 2011.

[11] S. Kozar and P. Kadera, “Integration of IEC 61499 with OPC UA”, *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1-7, 2016

[12] M. Schleipen, O. Sauer and J. Wang, “Semantic integration by means of a graphical OPC Unified Architecture (OPC-UA) information model designer for manufacturing execution systems”, 2010.

[13] R. Henben and M. Schleipen, “Interoperability between OPC UA and AutomationML”, *8th International Conference on Digital Enterprise Technology,* pp 297 – 304, 2012.

[14] M. Schleipen, E. Selyansky, R. Henssen and T. Bischoff, “Multi-level user and role concept for a secure plug-and-work based on OPC UA and AutomationML”, *IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1 - 4, 2015.

[15] F. Pauker, T. Fruhwirth, B. Kittl and W. Kastner, “A systematic approach to OPC UA information model design”, *49th CIRP Conference on Manufacturing Systems*, page 321 – 326, 2016

[16] D. Hastbacka, L. Barna, M. Karaila, Y. Liang, P. Tuominen and S. Kuikka, “Device Status Information Service Architecture for Condition Monitoring Using OPC UA”, *IEEE International Conference on Emerging Technology and Factory Automation (ETFA)*, pp. 1-7, 2014.

[17] S. Rohjans, M. Uslar and H. J. Appelrath, “OPC UA and CIM: Semantics for the smart grid”, *IEEE PES in Transmission and Distribution Conference and Exposition*, pp 1 – 8, 2010.

[18] S. Lehnhoff, W. Mahnke and S. Rohjans, “IEC 61850 based OPC UA Communication – The Future of Smart Grid Automation”, *17th Power Systems Computation Conference (PSCC'11)*, 2011.

[19] PLCOpen and OPC Foundation, “PLCopen and OPC Foundation: OPC UA Information Model for IEC 61131-3”, Release 1.00, March 24, 2010

[20] W. Dai, V. Vyatkin, J. Christensen, V. Dubinin, “Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Dynamic Reconfigurability”, *IEEE Transactions on Industrial Informatics*, Vol. 11, No. 3, pp 771 - 781, 2015.

[21] IEC 61131-3, Programmable controllers - Part 3: Programming languages, *International Standard*, Third Edition, 2013

[22] IEC 61499, Function Blocks, *International Standard*, Second Edition, 2012

[23] W. Mahnke, S. Leitner, and M. Damm, “OPC unified architecture”, *Springer Science & Business Media*, 327pps, 2009.

[24] W. Dai, C. Pang, V. Vyatkin, J. Christensen and X. Guan, “Discrete-Event-Based Deterministic Execution Semantics with Timestamps for Industrial Cyber-Physical Systems”, *IEEE Transactions on Systems, Man, Cybernetics: Systems*, in press, 2017.

[25] F. Palm, S. Grüner, J. Pfrommer, M. Graube and L. Urbas, “open62541–der offene OPC UA Stack”, 2014.