This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Jhunhunwala, Pranay; Atmojo, Udayanto Dwi; Vyatkin, Valeriy

Applying Skill-based Engineering using OPC-UA in Production System with a Digital Twin

Published in:

Proceedings of IEEE 30th International Symposium on Industrial Electronics, ISIE 2021

DOI:

[10.1109/ISIE45552.2021.9576342](https://doi.org/10.1109/ISIE45552.2021.9576342)

Published: 13/11/2021

Document Version

Peer reviewed version

Please cite the original version:

Jhunhunwala, P., Atmojo, U. D., & Vyatkin, V. (2021). Applying Skill-based Engineering using OPC-UA in Production System with a Digital Twin. In *Proceedings of IEEE 30th International Symposium on Industrial Electronics, ISIE 2021* (Proceedings of the IEEE International Symposium on Industrial Electronics). IEEE. <https://doi.org/10.1109/ISIE45552.2021.9576342>

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

© 2021 IEEE. This is the author's version of an article that has been published by IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Applying Skill-based Engineering using OPC-UA in Production System with a Digital Twin

Pranay Jhunjunwala*, Udayanto Dwi Atmojo*, Valeriy Vyatkin*[†]

*Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland

[†] Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden

Email: {pranay.jhunjunwala, udayanto.atmojo}@aalto.fi, vyatkin@ieee.org

Abstract—The increasing demand for flexibility in the production industry has brought the need for plug and produce systems, in which components from various vendors and suppliers can be easily integrated into the production plant and without a lot of effort on the ground-level these components can be controlled. This paper proposes a novel engineering approach, combining the IEC 61499 component-based organisation of software with high-level command interface, known as skills, communicated between the components via OPC-UA. The approach is illustrated on a laboratory-scale flexible assembly system, composed of pneumatic jacks, mechatronic conveyors, robots and AGV's. The proposed platform demonstrates the ease of improving flexibility in the production scenarios.

Index Terms—Skill, IEC 61499, OPC-UA

I. INTRODUCTION

Industrial manufacturing has traditionally aimed at mass quantity production. Production stations, machining tool, robots, and various types of mechanized or mechatronic actors in the factory floors are physically arranged usually in the form of fixed production line, where work-pieces may be put on the line at some points and then manipulated or processed along the way. Such a production line generally makes sense in the line of business where the product, e.g., has low price per unit. However, there is a growing interest of mass customization, where manufacturers need to cater for increasing demands from the customers for customized products.

Consequently, factory floors need to be able to flexibly adapt during operation, e.g., change in physical layouts, retrofits which may reflect that new production services associated to the new production actors need to be orchestrated, composed or decomposed to realize new production capabilities according to new production demands [1]. To accommodate flexible production in industry, a lot of research has been done in utilizing the Service Oriented Architecture (SOA) [2] in which automation control programs are designed in patterns in which various services from machining tools, robots or various mechatronic actors can be re-used to enhance the product driven manufacturing requirements of the industry.

In the skill-based engineering of automation systems software, the concept of skill was introduced for services provided by a machine or an autonomous part thereof [3]. Skills with basic functionality (e.g. controlling the motor of a conveyor) are composed into hierarchical structures to create composed

skills that can be used to execute complex procedures following the physical structure of the machine, composed of autonomous smart components [4]. Having a common platform/interface for all skill implementations will allow the users/consumers to easily visualise and control all the available basic, composite or complex skills [5].

Skill-based engineering has been proven to show better results in component exchange scenarios [6]. The concept of skills is seen as enabler of plug-and-produce approach to automation of flexible re-configurable production systems. The resulting increase in flexibility and configurability based on the use of skills has shown higher performance in the work [7]. Further research of the skill-based engineering has indicated the benefit of more focused engineering because the process designers and the PLC-programmers can practically work independently and get rid of the hassle of handling issues not specific to their expertise [8]. Somayeh et al in [9] discuss the possible challenges in Skill-based engineering and highlight the need of simulation models, factory topology, material flows to aid the standardization of skill-based engineering.

On the other hand, the IEC 61499 standard [10] was introduced to extend the capabilities of Programmable Logic Controllers in building flexible automation systems with decentralised decision making. The standard is getting traction with industry and is getting integrated with other important technologies, such as OPC-UA [11].

In this paper, a further extension of IEC 61499 with the skills is demonstrated aiming at the combination of benefits of both technologies. A case-study of a flexible assembly line with changeable layout, integrated with a mobile robot is investigated with a purpose of developing an automation architecture, combining the IEC 61499 distributed control with the Skills concept, communicating over OPC-UA. The desired feature is to enable peer to peer communication between the various actors involved in the decentralised decision making.

The rest of this paper is structured as follows: Section II describes a case-study system. Section III covers the control application developed in IEC 61499. Section IV introduces and describes the proposed OPC-UA Skill Profile. Section V concludes the paper and presents potential next steps.

II. CASE-STUDY SYSTEM AND AUTOMATION ARCHITECTURE

To illustrate the proposed architecture, a laboratory scale EnAS¹ assembly system was used. It is composed of a number of mechatronic units, such as conveyors combined with a group of stationary and mobile pneumatic actors that can manipulate workpieces and assemble them to produce final products. Each conveyor can only move in one direction and has photo eye object detection sensors to detect the presence of the workpiece on the conveyor at the particular position. The pneumatic jacks and sledges house various sensors to detect the position of the jack and sledge. The actual EnAS demonstrator system has been used for researching on automation concepts of future customized production in papers [1], [12], [13]. The system also includes ABB IRB14000 Yumi robot mounted on the SEIT100 Autonomous Guided Vehicle (AGV). A digital twin of EnAS has been developed with Visual Components (VC) modelling software. The developed digital twin can be seen in Fig. 1

Fig. 1: Visual Components Digital Twin of EnAS.

The proposed IT infrastructure of the skills based engineering consists of 3 parts shown in Fig. 2. Based on the SOA proposed in [2], the control program for the EnAS demonstrator has been developed with the IEC61499 standard using the NXT Studio software by NXTControl². Now as we can see the VC model and NXT Studio communicate over the inbuilt OPC-UA support provided by the respective software tools. In this case the server for OPC-UA is housed in the control device and VC behaves as a client.

Fig. 2: Automation System Architecture of EnAS.

The OPC-UA Server in NxtStudio has a limited capability and address space modification for the server wasn't possible on NxtStudio, hence an additional OPC-UA Server was created using the Prosys SDK, which was housing the OPC-UA Skill profile and the address space for the same. The skill server can be contacted by any OPC-UA client, UA Expert in our case, to get visualize the address space and take control via skills.

¹<https://www.energieautark.com/>

²<https://www.nxtcontrol.com/en/engineering/>

Fig. 3: Layered SOA architecture of the IEC 61499 control system of EnAS.

NxtStudio and the OPC-UA Skill Server would communicate over the TCP-IP protocol.

When only the control software would be commanding the digital twin, NxtStudio and VC exchange information over simple OPC-UA, but when we invoke skill control, the UA Client would send a request to the skill server over OPC-UA which would then command the control program over TCP-IP.

III. IEC 61499 CONTROL PROGRAM

The control software for the EnAS demonstrator has been developed based on the SOA paradigm using the function blocks of the IEC 61499 standard. Shown in Fig. 3 is the layered view of the SOA used in the application consisting. The bottom most layer i.e. the Core services layer holds the built-in functions used for communication of the software to the PLC's peripherals. The layer above this one is the user defined services layer in which the developed function blocks or function can utilise services provided by the Core Services layer. The second layer from top i.e. the Process layer has the information of the various processes controlled by the automation system eg: assembly operations over a work-piece by the help of mechatronic actors, such as jacks, or delivery of work-pieces to a certain conveyor using the delivery services. The top most layer i.e. the presentation layer takes control of the production sequences/scenarios/recipes.

The design of the core services layer is exemplified in Fig. 4. In it, built-functions such as E_MERGE and communication block SYMLINKMULTIVARSRC are used to map services to values of PLC outputs.

Fig. 4: An example of Core Service implementation.

Fig. 5: User Defined Services layer.

A part of the User Defined Services Layer implementation is shown in Fig. 5, where the blocks E_R_TRIG, E_DELAY, E_PERMIT are used to take the input from the core services block SYMLINKMULTIVARDEST to calculate the position of the jack which is then used by the services defined in the process layer.

The next layer i.e. the Process layer uses the information generated by the function blocks of the user-defined services layer to control the various functions of the system. The process layer in the control application for EnAS has two segments i.e. the delivery services and the production services. The delivery services FB, shown in Fig. 6a, is used to organise the various delivery sequences required by the production scenario's. The delivery services FB is a composite FB, housing various basic FBs which command the conveyors to start or stop based on the desired sequence. For example, when the delivery services FB receives the input command C3_to_C5, it invokes the movement sequence in which the work-piece will be carried from conveyor 3 to conveyor 5. The commands to the various conveyor blocks are sent via the adapter outputs seen in Fig. 6a.

The production services FB shown in Fig. 6b, is used to control the pneumatic jacks to either place a work-piece on the conveyor or remove a work-piece from the conveyor.

Fig. 6: Process Layer a) Delivery Services FB b) Production Services FB.

The final, the presentation layer, can consist of various production sequences as shown in Fig. 3. In our case, one production sequence block was implemented. Multiple instances of the same block were used to produce different products i.e. a green colored and a red colored product.

Fig. 7: Address Space a) Model b) Client.

IV. OPC-UA SKILL PROFILE

A major part of the project included application of the VDMA Skill companion specification for OPC-UA [14] for the purposes of implementing the customized production scenario.

This work introduces the Skill Execution Metamodel skill state machine, as well as the requirements of the OPC-UA parts and their implementation. These are discussed in the following subsections.

A. OPC-UA Address Space for Skill Profile

Based on [4], an address space shown in Fig. 7 is proposed for the Skill Profile. The main component is the SkillSet object that organizes all available skills on the server, which is modelled as a BaseObjectType. This is in line with the DeviceSet object of the OPC-UA for Devices standard. The SkillSet object has two components: the BasicSkills and ComposedSkills objects of the FolderType. These folders can both organize any number of skills of the respective nature.

For the skills themselves, a new type is defined, the SkillType for which the address space model is shown in Fig. 8. This type has three components: the optional variable ReturnData used by sensor skills to provide the actual sensor value, the SetStartParameter method which can be used to set an optional parameter for starting the skill and the StateMachine object.

A new type was created for it, the SkillStateMachineType. It is a subtype of the FiniteStateMachine and thus, has the variables CurrentState and LastTransition as required by the OPC-UA specification. These variables can have the value of one of the defined objects of type StateType and TransitionType, respectively. The methods lock, reset, start and stop are used to trigger the respective transitions of the state machine. A TransitionEvent is fired when the state machine transitions to a new state.

B. Implementation Structure

The software development was carried out using the IEC 61499 standard, with the NXTStudio IDE. Skill profile is based

Fig. 8: SkillType definition.

on the use of OPC-UA which is supported by NXTStudio but comes with a limitation in which the address space cannot be modified or customized. Therefore, a Java application using the Prosys OPC-UA SDK was developed. This application is used as the OPC-UA server for the skill profile. It manages the address space and can add and remove skills from the address space dynamically. The IEC 61499 function block communicates with this application through the plain text TCP commands. As such, the Java application can be seen as a bridge between the IEC 61499 solution and OPC-UA. In the following subsections, the NXTStudio and Java Implementations, are discussed, in sections IV-C and IV-D respectively.

C. IEC 61499 Implementation

The skill profile is implemented in a composite function block named TheSkill shown in Fig. 9. This FB corresponds to one skill in the UA address space and has been developed to be a universal fit to all the required skills.

The INIT event initializes the FB and opens the TCP connection to the Java server. It is worth mentioning that

Fig. 9: Skill FB Interface.

the INITO event will not be fired until a successful connection has been established. However, the skill is ready to be used locally i.e. without OPC-UA before this, so the local control application can operate without the Java application.

The skill can be controlled by the IEC 61499 program using the StartInternal and StopInternal events. The StartInternal event will first reset the skill (Locked → Idle) before starting it.

The EnableRemoteCtrl and DisableRemoteCtrl events are used to enable and disable the control of the skill through the OPC-UA methods. It defaults to the disabled state at the start-up, so remote control must explicitly be enabled before the skill can be controlled via OPC-UA.

The ErrorEvent event is connected to the skill state machine and will stop execution and lock the skill. This can be used as a system reset as the skill is reset to its original state. The NewData event is used to signal that updated data is available on the ReturnData variable. The ReturnData input is used if the skill is a sensor or similar, and must be able to return some data to the OPC-UA client. The NewData event should only be triggered if an updated value is present at ReturnData. The skill must also be in the Executing state for the data to update in the UA variable.

The output events Start, Stop and Reset are fired when the skill state machine transitions into the respective states. The Start event is associated with the StartParameter string that is an optional parameter set by the OPC-UA method SetStartParameter.

The internal structure of TheSkill FB is shown in Fig. 10. As can be seen there are two main components: the skill state machine and the SkillBridge FB detailed in section IV-C1 and IV-C2 respectively. The SkillBridge is the component that communicates with the OPC-UA server and provides the start, stop, etc., commands to the state machine. For every call of the UA methods stop, start, lock and reset, the SkillBridge FB expects to receive either the CallOK or CallDenied event. When the remote control is disabled, all the method call events are sent back to the CallDenied input event of the SkillBridge FB. Once the remote control is enabled, the events are sent to the corresponding event inputs on the skill state machine and to the CallOK input on the SkillBridge.

Fig. 10: TheSkill FB composition.

Fig. 11: Skill State Machine FB a) Interface b) ECC.

1) *Skill State Machine*: Fig. 11a shows the interface of the function block housing the skill state machine. The input events are the triggers for the respective transitions of the state machine. The current state as well as the last transition are outputted for use by the OPC-UA side of things. For each new state transition, the corresponding output event is fired.

To record the values for the current state of the skill i.e. locked or idle or executing, and also the previous executed state of the skill, the ECC of the SkillStateMachine function block shown in Fig. 11b was split into various states.

2) *Skill Bridge Function Block*: The purpose of the Skill-Bridge function block is to implement the TCP connection to the Skill Server. The internal structure of the FB is shown in Fig. 12. There are two main components: the TCPIO FB used for the TCP communication and the BridgeManager FB that implements the communication protocol.

As the BridgeManager contains all logic and data parsing, the internal structure is quite simple. On the TCPIO block an ENDCHAR of 10 is defined. This corresponds to LF, the new line character. This helps parsing the data, but some problems were identified when running the Skill Server on Windows vs Linux as the default line endings differ. This has been taken care of in the BridgeManager.

The ECC of the BridgeManager shown in Fig. 13 is implemented such that to the left is the initialization sequence. It begins by first sending the initialization command to the Skill Server and then parsing the returned status. If it fails, it tries to reconnect and reinitialize, otherwise it continues by sending an initial update command with the state and transition

Fig. 12: SkillBridge FB composition.

Fig. 13: Bridge Manage ECC.

to the server. If this was the first initialization, the INITO event is fired. After this the bridge is ready for normal operation and is waiting for action in the Waiting state. From here it can send updates to the Skill Server when the UPDATE event is triggered or receive data from the server on the REC event, triggered by the TCPIO FB. Once the data is received, it is parsed in the ParsePacket algorithm. This algorithm checks what method was called and then sets the right variable so that execution can continue to one of the Start, Stop, Reset or Lock states where the corresponding output event is fired. After this the bridge is waiting for the CallOK or CallDenied events to be able to send the call response back to the server. If an error occurs, the bridge manager re-initializes, i.e. reconnects to the Skill Server.

D. OPC-UA Skill Server Implementation

The functionality of the Skill Server can be described as a bridge between the SkillBridge function blocks in NXT and the OPC-UA address space. The SkillBridge FB sends commands to the Skill Server to create and update the UA representation of the skill and the Skill Server sends method calls to the SkillBridge FB when one of the UA methods is called.

Fig. 14 represents a the communication scenario between the SkillBridgeFB and the Skill Server.

Once the Skill Server is running it is waiting for TCP connections on a configurable port (default 4000). When the NXT solution is deployed/started, the TCPIO block of the SkillBridge FB will open a new connection to the server. For every incoming connection, the Skill Server will create a new instance of the SkillBridge Java class that is responsible for the connection to one skill FB in NXT. Once the connection is open, the SkillBridge function block will send an initialization command which includes the name and type of the skill. These are specified as constants on TheSkill FB. The name can be an arbitrary string but should not be too long. The type of the skill must be one of Basic, Composed, BasicSens or ComposedSens. The differences here are the folder in which the skill will end up in the address space and whether the skill will have the optional ReturnData variable enabled. Once the initialization command is received, the Java part of the bridge will create an instance of the SkillTypeNode and add it to the address space. The UA representation of the skill is now visible on the OPC-UA server. Next the SkillBridge FB will send an initial update command which includes the current state which the skill state machine is in, the last transition of the state

Fig. 14: Communication between Skill Server implemented with TCP Protocol.

machine and the string for the ReturnData variable, if present. This same update command is then sent every time the state machine changes state or when the skill is Executing and new data is available at the ReturnData variable.

When calling one of the UA methods of the skill, the Skill Server will send a method call command to the SkillBridge FB. This command includes the name of the method and if calling the start-method, then the start parameter set by the SetStartParameter method is also sent. After sending this command, the Skill Server is expecting to receive a response telling it if the call was successful or not. This response should be sent as fast as possible since the UA method call will not return before the response is received. This also means that if no response is received, the UA method call will fail with a timeout error. The main purpose of requiring this response is to implement the disabling of the remote control in NXT. If the response is negative, the UA method call will fail with BadRequestNotAllowed signalling that remote control is not allowed. The Skill Server will internally check the current state of the state machine when calling a method to check whether the call is allowed, e.g. calling start while Executing is not allowed. If the call is denied because of invalid state, the UA method call will fail with BadInvalidState.

V. CONCLUSIONS

This paper presented an automation architecture that facilitates flexibility in a production system with digital twin

components. The automation architecture utilized several software stacks, with the control software based on IEC 61499. Shown in the demo video³, the Skill-based engineering approach was adopted and implemented in OPC-UA standard to encapsulate individual functionality of the production system as skills, with different level of granularity.

Future works will include deployment of asset administration shell (AAS), adopting relevant OPC-UA companion specifications.

VI. ACKNOWLEDGEMENT

The authors would like to acknowledge Alexander Järvistö, Juri Voloskin, Oskari Järvi and Kevin Karlsson of Aalto University, for their effort and support during the development of the project.

REFERENCES

- [1] D. Drozdov, U. D. Atmojo, C. Pang, S. Patil, M. I. Ali, A. Tenhunen, T. Oksanen, K. Cheremetiev, and V. Vyatkin, "Utilizing software design patterns in product-driven manufacturing system: a case study," in *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*, pp. 301–312, Springer, 2019.
- [2] W. Dai, V. Vyatkin, J. H. Christensen, and V. N. Dubinin, "Bridging service-oriented architecture and iec 61499 for flexibility and interoperability," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 771–781, 2015.
- [3] J. Pfrommer, M. Schleipen, and J. Beyerer, "Pprs: Production skills and their relation to product, process, and resource," in *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, pp. 1–4, IEEE, 2013.
- [4] P. Zimmermann, E. Axmann, B. Brandenbourger, K. Dorofeev, A. Mankowski, and P. Zanini, "Skill-based engineering and control on field-device-level with opc ua," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1101–1108, IEEE, 2019.
- [5] K. Dorofeev and A. Zoitl, "Skill-based engineering approach using OPC UA programs," in *2018 IEEE 16th international conference on industrial informatics (INDIN)*, pp. 1098–1103, IEEE, 2018.
- [6] K. Dorofeev and M. Wenger, "Evaluating skill-based control architecture for flexible automation systems," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1077–1084, IEEE, 2019.
- [7] S. Profanter, A. Perzylo, M. Rickert, and A. Knoll, "A generic plug & produce system composed of semantic opc ua skills," *IEEE Open Journal of the Industrial Electronics Society*, 2021.
- [8] K. Dorofeev, "Skill-based engineering in industrial automation domain: skills modeling and orchestration," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, pp. 158–161, 2020.
- [9] S. Malakuti, J. Bock, M. Weser, P. Venet, P. Zimmermann, M. Wiegand, J. Grothoff, C. Wagner, and A. Bayha, "Challenges in skill-based engineering of industrial automation systems*," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 67–74, 2018.
- [10] IEC 61499, *Function blocks—Part 1: architecture, Second edition*. 2012.
- [11] "OPC UA specification, tech. rep.," 2012.
- [12] U. D. Atmojo, J. O. Blech, and V. Vyatkin, "A plug and produce-inspired approach in distributed control architecture: A flexible assembly line and product centric control example," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, pp. 271–277, IEEE, 2020.
- [13] U. D. Atmojo, K. Gulzar, V. Vyatkin, R. Ma, A. Hopsu, H. Makkonen, A. Korhonen, and L. T. Phu, "Distributed control architecture for dynamic reconfiguration: Flexible assembly line case study," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 690–695, IEEE, 2018.
- [14] *IAS-Companion Specification Proposal*. VDMA, 2018.

³Pranay Jhunjunwala, Udayanto Dwi Atmojo, & Valeriy Vyatkin. (2021). Applying Skill-based Engineering using OPC-UA in Production System with Digital Twin. Zenodo. <http://doi.org/10.5281/zenodo.4717677>