An OPC UA-Compliant Interface of Data Analytics Models for Interoperable Manufacturing Intelligence

Seung-Jun Shin

***Abstract*—The open platform communications unified architecture (OPC UA) has received attention as a standard for data interoperability in industries. In particular, OPC UA extends its applicability across various industrial sectors by publishing OPC UA companion specifications created in collaboration with other industrial consortiums. However, OPC UA is limited to ensure the interoperability of data analytics models because the relevant companion specifications have not been developed yet. OPC UA should be extended to implement such model interoperability so that machines seamlessly use and share data analytics models across the layers of manufacturing systems to predict and optimize their performance autonomously and collaboratively in terms of interoperable manufacturing intelligence. This article proposes an OPC UA-compliant interface for the exchange of predictive model markup language (PMML), a domain-independent standard for representing XML-based data analytics models. This article includes the design of mapping rules and OPC UA information models for the exchange between PMML and OPC UA, as well as the implementation of an OPC UA server-client prototype to publish and subscribe to OPC UA-compliant regression and neural network models which have been transformed from PMML-based models.**

***Index Terms*—Cyber–physical production systems, data analytics, manufacturing intelligence, model interoperability, open platform communications unified architecture (OPC UA), predictive model markup language (PMML).**

I. Introduction

CYBER–PHYSICAL Production Systems (CPPS) pursue the context-sensitive implementation of autonomous, cooperative, and responsive machines, products, and manufacturing systems within and across all levels of production [1]. Owing to the recognition of this pursuit, CPPS are considered to be cutting-edge enablers for manufacturing intelligence [1]. Manufacturing intelligence comprises the real-time and dynamic understanding, reasoning, planning, and management of manufacturing processes through the use of data analytics [2]; it largely relies on data-driven insights and foresights provided by smart machines and products, rather than the prior experience and intuition of humans. In this sense, data become a significant factor in manufacturing intelligence. Data should be collected, processed, transmitted, and exchanged without restriction so that machines and products can employ manufacturing intelligence autonomously and cooperatively. Thus, data interoperability is a prerequisite for manufacturing intelligence because it ensures a seamless exchange of data across heterogeneous devices and systems.

The open platform communications unified architecture (OPC UA) has recently received attention regarding data interoperability in the industrial automation realm [3]. The OPC UA is a data interface suitable for secure and reliable data exchange across a broad range of industries and has been formalized as the International Electrotechnical Commission 62541 standard [4]. The OPC UA provides unified and generic information modeling and transport mechanism across the layered hierarchies of manufacturing systems, thereby establishing information bridges between individual layers. Among many features of the OPC UA, its extensibility is the main factor that makes it such a feasible data interface to be implemented comprehensively and flexibly in various industrial domains by specifying how-to-design information models rather than stationary models. The OPC UA strategically adopts this extensible design philosophy through the release of domain-applicable companion specifications created in collaboration with industrial consortiums such as the International Society for Automation (ISA) and MTConnect.

Additionally, model interoperability is also crucial for manufacturing intelligence. Manufacturing intelligence can be achieved using the data-driven insights and foresights derived from data analytics, as described above. Data analytics mines and learns from data to produce data-driven operational knowledge, embodied by descriptive, diagnostic, predictive, and prescriptive models [5]. Such data analytics models endow machines and products with the ability to autonomously and cooperatively execute data-driven planning, execution, and control. Similar to the data themselves, data analytics models should be created, processed, transmitted, used, and exchanged without

Manuscript received January 29, 2020; revised April 24, 2020 and July 1, 2020; accepted September 11, 2020. Date of publication September 18, 2020; date of current version February 22, 2021. This work was supported by the Basic Research Program in Science and Engineering through the Ministry of Education of the Republic of Korea and the National Research Foundation under Grant NRF-2018R1D1A1B07047100. Paper no. TII-20-0418.

The author is with the Division of Interdisciplinary Industrial Studies, Hanyang University, Seoul 04763, South Korea (e-mail: sjshin@hanyang.ac.kr).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.3024628

restriction in a seamless model exchange environment. However, the OPC UA companion specifications ensuring the interoperability of data analytics models across heterogeneous devices and systems have not been published or released yet.

This article proposes an OPC UA-compliant interface for data analytics models to enable model interoperability for manufacturing intelligence. This OPC UA interface is designed and implemented to exchange the data analytics models formalized in the predictive model markup language (PMML) in the OPC UA environment. PMML is the standard developed by the Data Mining Group (DMG); it uses an XML-based language to represent not only descriptive and predictive models but also data pre/post-processing [6]. This article includes the following:

1) the definition of mapping rules for transforming PMML to the OPC UA;
2) the design of OPC UA domain-specific models for PMML, particularly, regression and neural network (NN) models;
3) the implementation of an OPC UA server-client prototype using OPC UA-compliant PMML models.

This prototype demonstrates that a virtual machine tool creates PMML-based predictive models of machining power, it can then encode such PMML models into an OPC UA-compliant form, and publish these OPC UA models in a server. Finally, a client subscribes to and decodes the OPC UA models for use.

The rest of this article is organized as follows. Section II addresses the motivation for the present study. Section III describes the related research on PMML and OPC UA. Section IV presents the model design for OPC UA-compliant PMML models. Section V describes the implementation of a prototype. Finally, Section VI concludes this article.

II. Interoperable Manufacturing Intelligence

This section presents the concept of interoperable manufacturing intelligence, which aims to interconnect CPPS with manufacturing systems through the interchange of OPC UA-compliant PMML models. This article is motivated by the need to develop a cross-domain interface between CPPS and manufacturing systems via the interchange of OPC UA-compliant PMML models.

CPPS create data analytics models for manufacturing intelligence, as described in Section I. Previous research in manufacturing intelligence has focused on creating descriptive, diagnostic, predictive, and prescriptive models to improve product quality and yield, reduce energy consumption, and perform predictive maintenance [7]; however, the exchange and sharing of such models has rarely been discussed [8]. Facilitating interoperability for data analytics models remains challenging owing to the lack of proper means to capture, structure, interchange, and encode/decode these models in traditional manufacturing systems.

Therefore, it is essential to implement interoperable manufacturing intelligence so that data analytics models can be exchanged and shared without restriction in manufacturing systems. The necessity for interoperable manufacturing intelligence can be considered in terms of the vertical and horizontal aspects of manufacturing systems. Suppose that machine tool systems

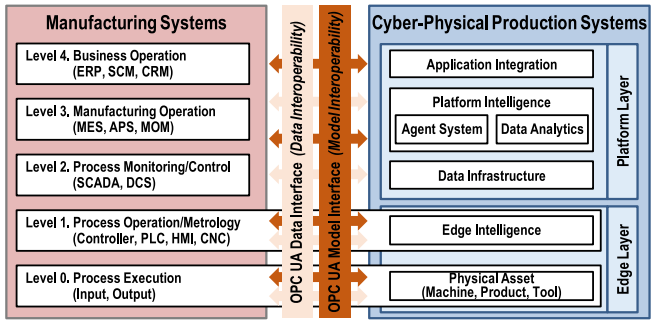

Fig. 1. Concepts of OPC UA interoperable manufacturing intelligence.

predict the energy consumed during their operations in order to diagnose, predict, and optimize energy efficiency in machining. In the vertical aspect, a machine tool creates its energy prediction models by learning data. It then needs to transmit these models to an upper system (e.g., manufacturing execution systems, platforms, cloud applications, and model repositories) mainly because of the memory limitations of the machine tool. In turn, the machine tool needs to receive energy prediction models from the upper system without restriction so that the machine can inform operators about anticipated energy values prior to actual machining or execute adaptive control for energy reduction through process parameter optimization. If model transmission and reception procedures are unfeasible or difficult, the machine tool will be unable to predict the energy consumption owing to the absence of energy prediction models. In the horizontal aspect, heterogeneous machine tools need to exchange and share energy prediction models through their mutual communication to predict the energy consumed in the entire production system. Similarly, these machines will be unable to predict the energy required for the production level if they fail to exchange energy prediction models compatibly.

Fig. 1 presents the concept of OPC UA-driven interoperable manufacturing intelligence. The left-hand side of the figure shows the ISA-95 framework for the vertical integration of manufacturing systems [9]. The OPC UA can work as a universal data interface for data interoperability within the ISA-95 framework, as discussed in Section I. The right-hand side represents the CPPS domain, which can comprise the edge and platform layers. In the edge layer, edge intelligence is empowered to physical assets for their online and adaptive control based on real-time data analytics. The platform layer pursues predictive and prescriptive planning and operations by batch or on-demand analytics at the system level. In the platform, the data infrastructure is a data repository that can handle large-sized and various data with high speed data processing, so-called big data. As the data analytics and the agent system are tightly integrated, the agent system endows platform intelligence with virtual assets, which mirror their physical ones onto the cyber world, to create, use and share data analytics models as individualized model creators and users. The application integration provides interfaces or services to connect with heterogeneous applications and thus allows for exchanging data analytics models on the application level. Such CPPS need to be converged with the ISA-95 framework through

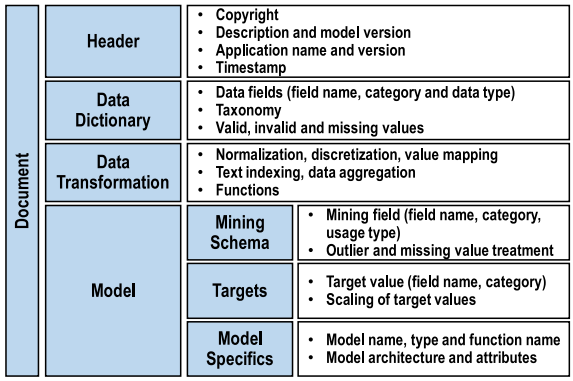

Fig. 2. Structure and contents of PMML (adapted from [6]).

a cross-domain interface to exchange the data-driven models created in the CPPS domain and employed in the manufacturing system domain. PMML is one of the most feasible representation languages for data-driven models because PMML specifies the structure and representation of data analytics models. This cross-domain model interface enables the transformation of data analytics models formalized in PMML into OPC UA-compliant models, ensuring OPC UA-based model interoperability. The model interface as a model supplier can be integrated with the OPC UA data interface, which acts as a data supplier for data analytics because these two interfaces identically use the OPC UA. Therefore, the model interface enables the implementation of interoperable manufacturing intelligence complementarily with the OPC UA data interface. The purpose of the present study is to develop this model interface; this task is divided into model design (see Section IV) and function implementation (see Section V).

III. PMML AND OPC UA

The input used in the present study is PMML, and the output is OPC UA. Thus, an understanding of their respective structures and contents is required. Section III-A describes the structure of PMML and reviews previous studies in manufacturing. Section III-B describes the structure of OPC UA and the relevant research.

A. PMML

PMML provides an open standard for representing data pre/post-processing as well as data mining models such as regression, support vector machine (SVM), and NN. The PMML structure is represented by the XML schema, which specifies the data structure, representation, and relations in an XML Schema Definition file [10]. PMML is designed for generic uses; it enables the exchange of data analytics models between data analysis tools such as R and KNIME [6]. PMML is useful for processing data analytics models in computer. As data analytics models are typically expressed as mathematical formulae, PMML can explicitly represent and formalize such mathematical models using structural and textual languages.

Fig. 2 illustrates the major elements of the PMML structure. A PMML document is a root element, which is composed of header, data dictionary, data transformation, model, mining schema, and target elements [10]. The "model" element identifies a selection of single or multiple models and contains the mining schema, targets, and model specifics. "Mining schema" specifies the data fields that need to be provided when applying the model. "Model specifics" defines the structure and representation of the model(s); this varies with the model(s) selected due to their different contents and structures.

The deployment of PMML in manufacturing is increasing as data analytics provides a data-driven method for implementing manufacturing intelligence. Accordingly, major research streams can be categorized in terms of model representation, extension, and pioneering. Model representation indicates when PMML is adopted as a representation format for data analytics models. O'Donovan *et al.* [11] used SVM models to identify component issues in an air handling unit and encoded these models into PMML files. Lechevalier *et al.* [12] developed a semi-automatic method for predictive model generation and encoded NN models into PMML for sharing between PMML-compliant tools. O'Donovan *et al.* [13] presented a fog computing environment that embedded PMML-encoded machine learning models in factory operations. Model extension refers to the stage of adding information to the original PMML to increase its usability and traceability. Lechevalier *et al.* [14] proposed a structural metamodel for storing and retrieving PMML-based NN models in and from databases. Zhang *et al.* [8] developed an information model that connected data analytics models with product, process plan, and equipment information to increase model traceability. Model pioneering deals with the verification and validation of new PMML models in manufacturing as one of test fields and thus enriches the comprehensiveness of PMML. Park *et al.* [15] and Nannapaneni *et al.* [16] developed PMML schema for Gaussian process regression and Bayesian networks models, respectively, by demonstrating their feasibility for energy prediction of a machining or welding process. These models were officially included from PMML version 4.3.

Previous studies have facilitated the employment of PMML for sharing the data analytics models used to diagnose, predict, and optimize the target KPIs of various processes; they have subsequently led to improvements in the usability, traceability, and comprehensiveness of PMML. However, their use of PMML has depended on the file transactions of data analysis tools. Otherwise, they create and parse PMML files by implementing their own encoders and decoders, which are independent of and irrelevant to interoperability in manufacturing. Therefore, it remains crucial to develop methods for exchanging PMML in a unique and interoperable manner across hierarchical manufacturing systems and heterogeneous devices.

B. OPC UA

The OPC UA uses XML, binary, and Java Script Object Notation (JSON) as its representation languages [4]. In the system view, the OPC UA is embodied based on the server–client

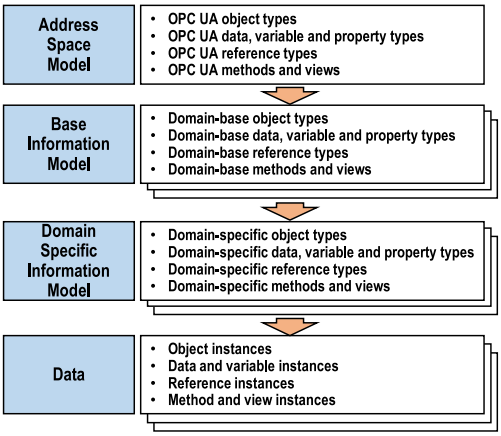

Fig. 3. Structure and content of the OPC UA information model.

concept, wherein a server publishes and a client subscribes to OPC UA-based data content.

In terms of information, the structure of the OPC UA consists of address space model, base information model, domain-specific information model, and data layers [17], as visualized in Fig. 3. The address space model (ASM) is a metamodel for OPC UA-compliant information modeling and defines basic nodes and relations as well as methods and constraints. The base information model (BIM) inherits the ASM and is the container-model for defining domain-based objects, variables, properties, references, and methods. The domain-specific information model (DSIM) inherits the BIM; it specifies the detailed information model dedicated to the target domain. The data layer expresses the real data instances that fit into the DSIM. This modeling approach conforms to the metamodeling that enables the extensibility of the OPC UA, as described in Section I. Domain experts can develop their OPC UA-compliant models by conforming the DSIM to the ASM.

One challenge is facilitating the data exchange required for vertical integration over the hierarchy of manufacturing systems as well as horizontal integration across heterogeneous field devices [18]. Accordingly, many data and communication protocols, such as the simple network management protocol, lightweight directory access protocol, native web technologies, and OPC UA have been competitively or complementarily applied in manufacturing [18]. Of these, the OPC UA has emerged as an important standard since the Reference Architecture Model Industry 4.0 recommended the OPC UA as an implementer for communication layers in CPPS [19].

Accordingly, leading studies have been performed to apply the OPC UA to CPPS. Schlechtendahl *et al.* [20] proposed an architecture in which production systems are integrated into CPPS through an OPC UA-driven communication gateway and information server. Grüner *et al.* [21] developed the RESTful architecture, which incorporated OPC UA for industrial communication using resource-oriented information models in CPPS. Schleipen *et al.* [22] implemented a unique OPC UA data hub for quality defect tracking, monitoring, control, and CPPS orchestration in multiple programming languages and operating systems. Givehchi *et al.* [23] implemented an interoperability layer for migrating legacy systems to CPPS through integration of ISA-95 and OPC UA. Cavalieri *et al.* [24] developed an REST-based OPC UA platform to provide access to multiple OPC UA servers from the web in a resource-oriented manner.

OPC UA-compliant information modeling is another research stream that has considered the extensibility of OPC UA. Some studies have developed OPC UA-interoperable information models for the interchange with existing information models and standards. Rohjans *et al.* [25] designed the OPC UA-compliant common information model to build a semantic web service for smart grid. Lee *et al.* [26] proposed a method to convert unified modeling language to OPC UA for efficient software engineering. Katti *et al.* [27] remodeled the OWL ontology language for services with OPC UA compliance to allow for flexible orchestration plans. Liu *et al.* [3] developed an OPC UA information model for machine tools with consideration of the interoperability between OPC UA and MTConnect. However, thus far, no studies have investigated the integration of OPC UA and PMML. Previous studies have primarily focused on data interoperability between CPPS and legacy manufacturing systems or information modeling for harmonizing existing models and standards with the OPC UA. Thus, they are limited to ensure model interoperability by allowing data analytics models to be shared between manufacturing systems and CPPS. This integration is significant for implementing interoperable manufacturing intelligence by bridging the gap between manufacturing systems and CPPS.

IV. Model Design

This section presents the design of the OPC UA-compliant PMML models. In Section IV-A, mapping rules are identified to convert PMML (version 4.3) into OPC UA because OPC UA and PMML have different structures and contents over disparate domains. In Section IV-B, the BIM and DSIM designated to PMML are designed based on the mapping rules; this is followed by specific details concerning the regression and NN models. The regression and NN models are chosen among 17 models because they are commonly used for predictive modeling in the manufacturing domain [7].

A. Mapping Rules

XML is reasonable for use as the representation language because the current PMML only supports XML; on the other hand, the OPC UA supports XML, binary, and JSON. Both PMML and the OPC UA build upon object-oriented technology, which comprises nodes and references; thus, their mapping can be feasible and effective. However, one-to-one mapping is partially available because some nodes and references in the PMML schema do not align exactly with those of the OPC UA ASM. Thus, semantic-based mapping is also necessary. In other words, one-to-one mapping is applied when the nodes and references of PMML are symmetrical with those of the OPC UA, whereas manual mapping is applied using semantics and domain knowledge when the nodes and references of PMML are asymmetric or semantically different from those of the OPC

TABLE I
MAPPING TABLE FOR CONVERTING PMML TO OPC UA

Group	PMML	OPC UA	Description
Object	Element	Object Type	- Type definition: object type - Instance: object
	Document	'Document'	- Top level object type
	MODEL-ELEMENT	'Model element'	- Abstract object type - Contain common elements for data analytics models
Variable	Attribute	Variable Type (Data variable)	- Type definition: variable type - Instance: data variable (when attributes contain dynamic and changeable values)
	Attribute	Variable Type (Property)	- Type definition: variable type - Instance: property (when attributes contain static and unchangeable values)
Data type	Attribute type	Built-in data type	- Type definition: built-in data type (when attribute types use typical data types)
	Attribute type	User-defined data type	- Type definition: user-defined data type (when attribute types are specific for PMML) - Inherit built-in data types
Reference	Ref	Reference Type	- Type definition: HasSubtype, HasComponent, HasProperty, HasTypeDefinition or Organize (depending on semantics)
Modeling rule	use= "required"	'Mandatory'	- Must be instantiated
	use= "optional"	'Optional'	- Allow optional instantiation
	minOccurs/ maxOccurs	'Placeholder'	- Mandatory or optional place holder (when these attribute types are meaningful, the mandatory rule is applied)

UA. Table I presents a mapping table for converting the PMML schema into the OPC UA information model.

1) Object: In the OPC UA ASM, "ObjectType" and "Object" are used for object type definitions and object instances, respectively [17]. Thus, the "element" in PMML can be one-to-one mapped to the "ObjectType" of OPC UA because both specify node classes. "Document" in PMML——the root node——is mapped to the "Document" type of OPC UA. In the original PMML schema, the "MODEL-ELEMENT" lists a group of available data analytics models and allows only for the choice of single or multiple models from the model list. We create an abstract "model element" object type to bind the mining schema, output, local transformation, model statistics, explanation, and verification elements. These are common elements that every data analytics model uses and may contain.

2) Attribute: In the OPC UA, "VariableType" is used for the type definitions of variables, whereas "Data Variable" or "Property" is used selectively for instances of variables. "Data Variable" is used to represent the data of an object; however, "Property" is used to represent the characteristics of an object or data variable [4]. This separation is not easy [17]. The "Attribute" of PMML can be one-to-one mapped to the "VariableType" of OPC UA; however, its instances can be separated into "Data Variable" or "Property." We use "Data Variable" for instances of dynamic and changeable attribute values (e.g., coefficients in a model) and "Property" for instances of static and unchangeable values (e.g., names or units in a model).

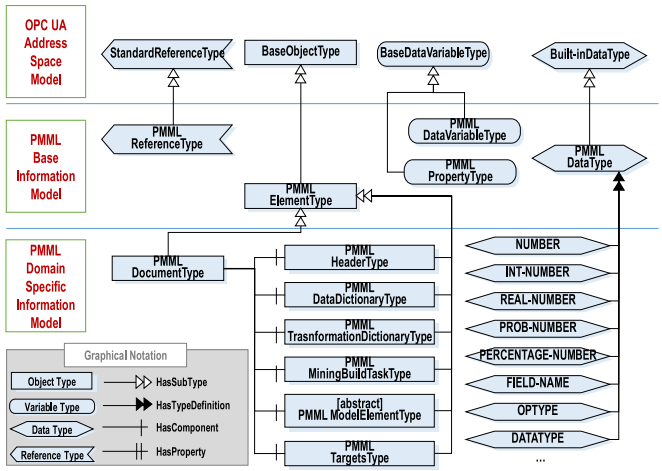

Fig. 4. PMML base and domain specific information models (top-level).

3) Data Type: Every attribute in PMML accompanies its data type. Typical data types (e.g., string, integer, double, and boolean) in PMML can belong to the data types built into the OPC UA. Meanwhile, PMML also defines its domain-specific data types as well as enumeration types. We create user-defined data types for such domain-specific data and enumeration types with inheritance of the OPC UA built-in data types.

4) Reference: The "Ref" of PMML is one-to-one mapped to the "ReferenceType" of the OPC UA. It can be divided into "HasSubtype," "HasComponent," "HasProperty," "HasTypeDefinition," or "Organize" depending on the relational semantics. If two elements relate to a super- or subtype, these are tied into "HasSubtype." If two elements have a part-to-whole relationship, "HasComponent" is employed. "HasProperty" only works for property instances. "HasTypeDefinition" is used to bind an object or data variable to its object type or variable type [4].

5) Modeling Rule: The modeling rules in the OPC UA specify instance declaration with respect to instances of "ObjectType" and "VariableType." These rules are mainly used to declare naming rules (mandatory or optional), constraints, and cardinality restriction [17]. PMML contains "required" and "optional" rules for attributes, which can be one-to-one mapped to "Mandatory" and "Optional" in the OPC UA modeling rules, respectively. "MinOccurs" and "maxOccurs" can be used to define the cardinality restrictions when they are semantically meaningful.

B. PMML Base and Specific Information Models

We design a BIM and DSIM for PMML based on the mapping rules described in Section IV-A. Fig. 4 depicts the top-level BIM and DSIM created in compliance with the OPC UA structure and notation. In the BIM, the types of elements, references, data variables, properties, and data are inherited from their corresponding basic types in the OPC UA ASM. "Element type"

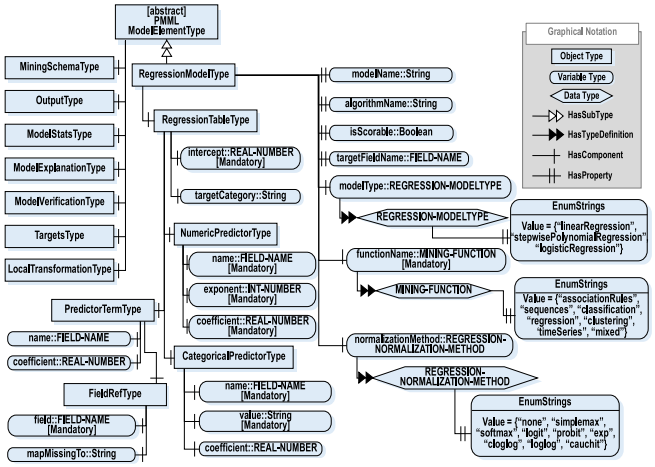

Fig. 5. Domain-specific information model of a regression model.

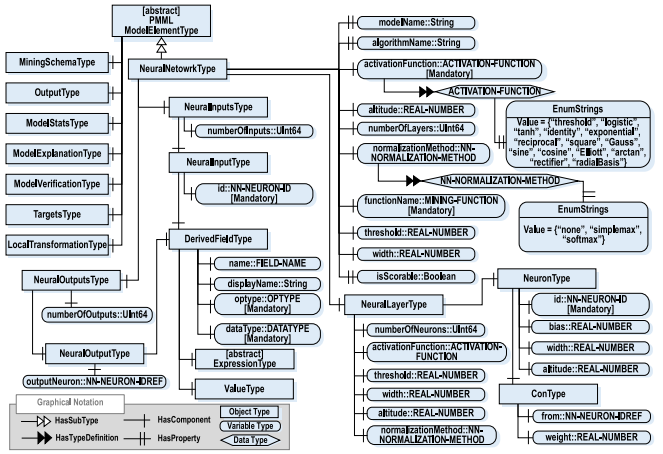

Fig. 6. Domain-specific information model for a neural network model.

is a meta-object type that represents every PMML element. In the DSIM, "document type" is the root element that consists of the subcomponents. PMML-specific data types are identified through inheritance of built-in data types in the OPC UA ASM. The following describes the DSIM specifically for regression and NN models.

1) Regression Model: Fig. 5 depicts the DSIM of a regression model, which is inherited from the abstract "model element" type. The "model type" is identified through the choice of a model type from the enumeration values. "Function name" specifies a mining function for which a value must be chosen from the values of the "MINING-FUNCTION" enumeration type. Suppose that "linearRegression" is chosen as the model type. Equation (1) expresses a univariate linear regression formula. The following items describe the mapping of the attributes of the regression equation to their associated variable types in Fig. 5. These attributes are mandatory because they use the "required" modeling rule and must be instantiated so as to fulfill (1). The details of variables x and y are declared in the "mining field type" inside "mining schema type"

$$\mathrm{y} = b_0 + b_1 x + \ldots + b_k x^k \tag{1}$$

where

1) b_0 (intercept): "intercept" in "regression table type";
2) x (input variable): "name" in "numeric predictor type";
3) k (exponent): "exponent" in "numeric predictor type";
4) b_k (coefficient): "coefficient" in "numeric predictor type";
5) y (output): "name" in "output field type" of "output type."

2) Neural Network Model: Fig. 6 illustrates the DSIM of an NN model, which is also inherited from the abstract "model element" type. This DSIM differs from that of the regression model by its structure. The "model element" abstract type contains the common elements including mining schema, output (a set of result values returned from models), model statistics (variable statistics), model explanation (explanatory details of models), model verification (dataset and result for verifying models), targets (properties of the result values), and local transformation (normalization, discretization, value mapping, text indexing, functions, and aggregation).

Equation (2) describes an output neuron (o_i), which is computed as a simple weighted summation over the responses of the hidden neurons to given input variables [28]. In Fig. 6, each neural layer is specified in the "neural layer type" and the number of layers is identified in the "number of layers" in "neural network type." Each "neuron type" must have its own ID and may have a bias in addition to the weights from the "con type," which defines the individual neuron IDs in the previous layer. "Activation function" is defined in both the "neural network type" and "neural layer type" for a selective option in the model or layer level

$$o_i\ (x_n) = \sum_{j=1}^{M} w_{ij}\theta_j\ (x_n) + w_{\mathrm{bias},i} \tag{2}$$

where

1) x_n (input variable): "name" in "derived field type";
2) w_{ij} (weight): "weight" in "con type";
3) θ_j (activation function): "activation function" in "neural network type" or "neural layer type";
4) $w_{\mathrm{bias},i}$ (bias): "bias" in "neuron type";
5) M (number of neurons): "number of neurons" in "neural layer type."

V. Implementation

This section presents a system architecture and a prototype implementation to design the functions for the proposed model interface and demonstrate its feasibility and effectiveness. Sections V-A, V-B, V-C, and V-D describe the architecture, scenario, implementation, and discussion, respectively.

A. Architecture

Fig. 7 depicts a system architecture, which pursues total integration with the OPC UA for data and model interoperability. The scheme of the architecture is a chained server pattern, where Client 1 externally communicates with Server 1, and Server

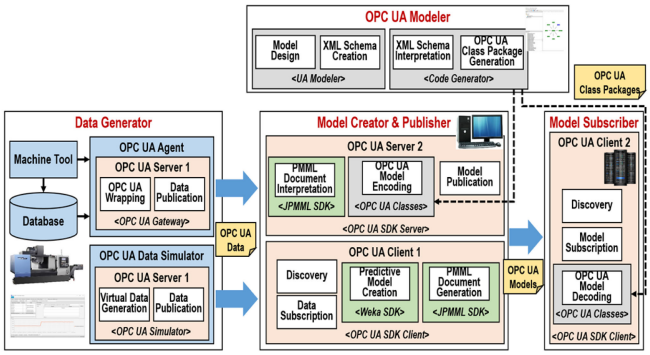

Fig. 7. Implementation architecture.

2 embeds Client 1 and externally exchanges messages with Client 2.

1) OPC UA Modeler: The PMML DSIM should be pre-modeled to convert the graphical notations (see Figs. 4–6) into explicit programmable codes. A modeling tool can be used to perform this conversion, as this tool provides graphical design, model specification, and XML schema file generation [29]. Once the PMML DSIM has been coded and compiled using the tool, an XML file that defines the schema of the DSIM under OPC UA compliance is automatically created. Next, this XML file is interpreted to automatically generate OPC UA-compliant PMML Java class packages.

2) Data Generator (Server 1): Data can be generated and obtained from real machines, databases, or virtual simulators. Machines can generate their machine-monitoring data dynamically, and these data can be serially wrapped by an OPC UA agent [this agent can be implemented using OPC UA Application Programming Interfaces or Software Development Kit (SDK)] to publish OPC UA data content. Databases that store machine-monitoring data can retrieve tabular datasets (e.g., comma-separated value files) and send these structured datasets to the OPC UA agent for publishing OPC UA data content. Additionally, virtual simulators can be used to generate virtual time-series and OPC UA-compliant data automatically based on equations defined by the user [30].

3) Model Creator and Publisher (Client 1 and Server 2): This module comprises Client 1 and Server 2. Client 1 subscribes to the machine-monitoring data generated by the data generator. This client creates data analytics models and produces their corresponding PMML documents. The discovery and connection of the server-client can be achieved through the OPC UA transport mechanism (this is beyond the scope of the present work). Next, Server 2 interprets and converts PMML documents to PMML objects. Server 2 then encodes these object instances to OPC UA-compliant objects with the use of the OPC UA-compliant PMML classes. Finally, Server 2 publishes these OPC UA objects.

4) Model Subscriber (Client 2): This module subscribes to and decodes the OPC UA-compliant PMML objects for accumulation and utilization in the subsequent stages. Client 2 connects with Server 2 using the authentication and security in the OPC UA transport mechanism.

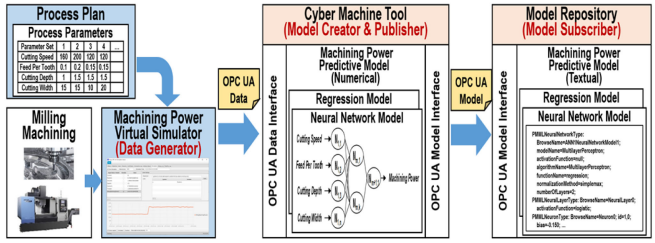

Fig. 8. Case study scenario.

TABLE II
FACTOR AND LEVEL SETTING FOR DATA GENERATION

Factor	Unit	Level		
		-1	0	1
Cutting speed	m/min	120	160	200
Feed per tooth	mm/rev	0.1	0.15	0.2
Cutting depth	mm	1	1.5	2
Cutting width	mm	10	15	20

B. Case Study

In the metal cutting industry, energy-efficient machining is critical for reducing the energy consumed during the operation of turning or milling machine tools [31]. Accordingly, machining power is an important metric because the energy consumption is obtained from the integral of the machining power over time. Predicting the machining power during the process planning phase enables the determination of process parameters (cutting speed, feed per tooth, cutting depth, and width) for energy-efficient machining [31]. Thus, creating predictive models for machining power is necessary to determine the numerical relationship between the process parameters (input variables) and machining power (an output variable). Fig. 8 shows a scenario for this case study. In this scenario, we create predictive models for machining power (hereafter, power predictive models) in a milling machine tool and then share these models with another system.

In Fig. 8, a milling machine tool generates machining power data during operation. Here, a virtual simulator is used to generate virtual machining power data. Equation (3) expresses the formula for machining power [32]. Machining power (P_m) comprises the cutting power (P_c) and the machine's turned-ON power (P_t). We assume that the workpiece material is mild steel with a tensile strength of 520 MPa and that the machine consumes 1000 W of turned-ON power. Under the Box–Behnken design of experiments [33], as presented in Table II, the four process parameters are set as factors, their three levels are structured, and 27 trials are performed. Each trial produces machining power data tuples during 30 s

$$P_m(w) = P_c + P_t = \frac{(a_p a_e f_z v_c) K_c z}{60\pi D_c \eta} + P_t \qquad (3)$$

where a_p (mm): cutting depth, a_e (mm): cutting width, f_z (mm/teeth): feed per tooth, v_c (m/min): cutting speed, K_c (MPa): specific cutting force (= 1950, 2075, or 2200 corresponding to feed per tooth values of 0.1, 0.15, or 0.2, respectively), z: the

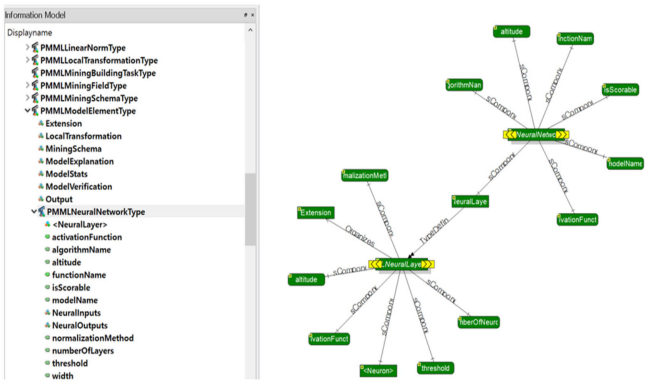

Fig. 9. Screen shot of the UA modeler.

```
<UAObjectType NodeId="ns=1;i=1059" BrowseName="1:NeuralNetworkType">
    <DisplayName>NeuralNetworkType</DisplayName>
    <References>
      <Reference ReferenceType="HasComponent">ns=1;i=5082</Reference>
      <Reference ReferenceType="HasComponent">ns=1;i=6174</Reference>
      <Reference ReferenceType="HasComponent">ns=1;i=6173</Reference>
      <Reference ReferenceType="HasComponent">ns=1;i=6177</Reference> ... </References> </UAObjectType>
  <UAObject SymbolicName="NeuralLayer" ParentNodeId="ns=1;i=1059" NodeId="ns=1;i=5082"
   BrowseName="1:&lt;NeuralLayer&gt;">
    <DisplayName>&lt;NeuralLayer&gt;</DisplayName>
    <References> <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=1059</Reference>
      <Reference ReferenceType="HasModellingRule">i=11508</Reference>
      <Reference ReferenceType="HasTypeDefinition">ns=1;i=1061</Reference> </References> </UAObject>
  <UAVariable DataType="ACTIVATIONFUNCTION" ParentNodeId="ns=1;i=1059" NodeId="ns=1;i=6174"
   BrowseName="1:activationFunction" AccessLevel="3">
    <DisplayName>activationFunction</DisplayName>
    <References> <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=1059</Reference>
      <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
      <Reference ReferenceType="HasModellingRule">i=78</Reference> </References> </UAVariable>
  <UAVariable DataType="String" ParentNodeId="ns=1;i=1059" NodeId="ns=1;i=6173" BrowseName="1:algorithmName"
   AccessLevel="3">
    <DisplayName>algorithmName</DisplayName>
    <References> <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=1059</Reference>
      <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
      <Reference ReferenceType="HasModellingRule">i=80</Reference> </References> </UAVariable>
  <UAVariable DataType="REALNUMBER" ParentNodeId="ns=1;i=1059" NodeId="ns=1;i=6177" BrowseName="1:altitude"
   AccessLevel="3">
    <DisplayName>altitude</DisplayName>
    <References> <Reference ReferenceType="HasComponent" IsForward="false">ns=1;i=1059</Reference>
      <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
      <Reference ReferenceType="HasModellingRule">i=80</Reference> </References> </UAVariable> ...
```

Fig. 10. Example of the XML schema for PMML domain-specific models.

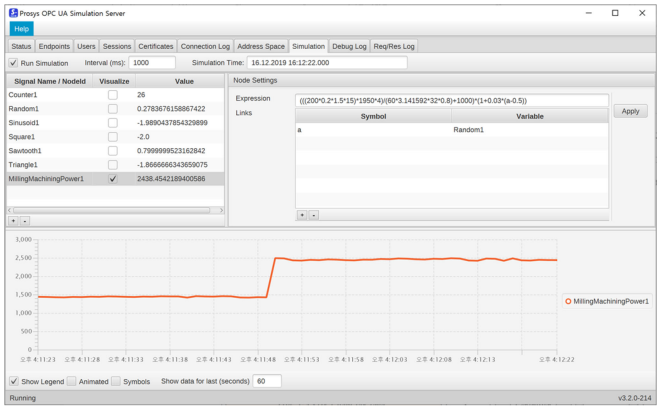

Fig. 11. Screen shot of the OPC UA Simulation Server.

number of tool teeth ($= 4$), D_c (mm): tool diameter ($= 32$), and η: machine's energy efficiency ($= 0.8$).

Then, the machining power data are combined with the corresponding process parameters to generate an x–y pairwise training dataset. In turn, the cyber machine tool, which corresponds to Model Creator and Publisher in Fig. 7, learns the training dataset and creates two different types of power predictive models: regression and NN. The machine encodes these models using PMML and then converts the PMML models into the corresponding OPC UA-compliant models. These OPC UA models are transmitted to a model repository, which corresponds to Model Subscriber. This repository decodes and stores the OPC UA models for future use (refer to Section II for the utilization scenarios).

C. Implementation Result

Based on the system architecture depicted in Fig. 7, we implement a prototype system. The software for this implementation includes Eclipse Oxygen for Java programming, Weka SDK (version 3.8) for predictive modeling, JPMML SDK for the PMML document handling, UA Modeler (version 1.6.2) for the OPC UA modeling, UA Code Generator (version 4.0.0) for the OPC UA class code generation, Prosys OPC UA simulation server (version 3.2.0) as the data simulator, Prosys OPC UA SDK for Java (version 4.0.0) as the server and client. The server–client is implemented on the localhost, and the hardware resource is a single personal computer with a Windows 10 Pro operating system, Intel Core i7-8700 processor, and 32 GB of memory.

1) OPC UA Modeler: We use UA Modeler to create the PMML DSIM through textual inputs in the designated forms. This tool allows for visualizing the DSIM in graphical and hierarchical notations, as shown in Fig. 9. We also use this tool to create the XML file that defines the schema of the DSIM, as shown in Fig. 10. We use UA Code Generator to generate OPC UA-compliant PMML Java class packages. This tool produces Java class and interface files that include getters and setters of object, variable, and reference types as well as data types [34], thus reducing the programming load.

2) Virtual Data Simulator (Server 1): The data simulator generates virtual machining power data, as illustrated in Fig. 11. We apply (3) with a $\pm 1.5\%$ random deviation to generate virtual machining power data at intervals of 1 s. We change the process parameters at every 30 s based on the factorial trials presented in Table II.

3) Cyber Machine Tool (Client 1 and Server 2): We implement Client 1 and Server 2 using OPC UA SDK. Client 1 receives machining power data from the data simulator and combines the data with their associated process parameter sets to generate a training dataset. Then, Client 1 creates the regression and NN-based power predictive models using Weka SDK, an open-source data mining solution [35]. The regression model uses a linear regression function, as expressed in (1); the NN model applies a multilayer perceptron function [four neurons in the input layer, two neurons in the hidden layer, one neuron in the output layer, a "logistic" activation function in the input and hidden layers, and an "identity" activation function in the output layer; refer to (2) for the NN function]. The regression model results in a root mean square error (RMSE) of 77.278 W and a relative absolute error (RAE) of 25.6376%, whereas the NN model records an RMSE of 31.456 W and RAE of 7.9211%. Once such power predictive models have been created, Client 1 produces the two PMML documents (regression and NN models) via JPMML SDK, which produces and interprets

```
<PMML xmlns="http://www.dmg.org/PMML-4_3" xmlns:data="http://jpmml.org/jpmml-model/InlineTable" version="4.3">
  <Header description="PMMLANNModelbyWeka">
<DataDictionary numberOfFields="5">
    <DataField name="CuttingSpeed" optype="continuous" dataType="double"/>
    <DataField name="FeedPerTooth" optype="continuous" dataType="double"/>
    <DataField name="CuttingDepth" optype="continuous" dataType="double"/>
    <DataField name="CuttingWidth" optype="continuous" dataType="double"/>
    <DataField name="MachiningPower" optype="continuous" dataType="double"/> </DataDictionary>
  <NeuralNetwork modelName="MultilayerPerceptron" functionName="regression" algorithmName="MultilayerPerceptron"
   normalizationMethod="simplemax" numberOfLayers="2">
    <MiningSchema>
      <MiningField name="CuttingSpeed" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="FeedPerTooth" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="CuttingDepth" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="CuttingWidth" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="MachiningPower" usageType="predicted" invalidValueTreatment="asIs"/> </MiningSchema>
    <Output> <OutputField name="MachiningPower" optype="continuous" dataType="double" feature="predictedValue"
      value="MachiningPower"/> </Output>
    <NeuralInputs numberOfInputs="4">
      <NeuralInput id="0,0"> <DerivedField name="CuttingSpeed" optype="continuous" dataType="double"/> </NeuralInput>
      <NeuralInput id="0,1"> <DerivedField name="FeedPerTooth" optype="continuous" dataType="double"/> </NeuralInput>
      <NeuralInput id="0,2"> <DerivedField name="CuttingDepth" optype="continuous" dataType="double"/> </NeuralInput>
      <NeuralInput id="0,3"> <DerivedField name="CuttingWidth" optype="continuous" dataType="double"/> </NeuralInput>
    </NeuralInputs>
    <NeuralLayer activationFunction="logistic"> <Neuron id="1,0" bias="-0.15038276122101782">
        <Con from="0,0" weight="0.4940730222511675"/> <Con from="0,1" weight="0.6474712892546692"/>
        <Con from="0,2" weight="1.2872593805539925"/> <Con from="0,3" weight="1.375440049462113"/> </Neuron> ...
```

Fig. 12. Example of a PMML document for a neural network model.

```
HeaderType: BrowseName=ANN1Header1; description=PMMLANNModelbyWeka;
ExtensionType: BrowseName=ANN1Extension1; name=Machine Tool; extender=null; value=null;
ApplicationType: BrowseName=ANN1Application1; name=JPMML; version=1.4.9;
DataDictionaryType: BrowseName=ANN1DataDictionary1; numberOfFields=5;
DataFieldType: BrowseName=ANN1DataField0; name=CuttingSpeed; dataType=Double;optype=continuous;
DataFieldType: BrowseName=ANN1DataField1; name=FeedPerTooth; dataType=Double;optype=continuous;
DataFieldType: BrowseName=ANN1DataField2; name=CuttingDepth; dataType=Double;optype=continuous;
DataFieldType: BrowseName=ANN1DataField3; name=CuttingWidth; dataType=Double; optype=continuous;
DataFieldType: BrowseName=ANN1DataField4; name=MachiningPower; dataType=Double;optype=continuous;
ModelElementType: BrowseName=ANN1ModelElement1;
NeuralNetworkType: BrowseName=ANN1NeuralNetworkModel1; modelName=MultilayerPerceptron; activationFunction=null;
algorithmName=MultilayerPerceptron; functionName=regression; normalizationMethod=simplemax; numberOfLayers=2;
MiningSchemaType: BrowseName=ANN1MiningSchema1;
MiningFieldType: BrowseName=ANN1MiningField0; name=CuttingSpeed; invalideValueTreatment=asIs; usageType=active;
MiningFieldType: BrowseName=ANN1MiningField1; name=FeedPerTooth; invalideValueTreatment=asIs; usageType=active;
MiningFieldType: BrowseName=ANN1MiningField2; name=CuttingDepth; invalideValueTreatment=asIs; usageType=active;
MiningFieldType: BrowseName=ANN1MiningField3; name=CuttingWidth; invalideValueTreatment=asIs; usageType=active;
MiningFieldType: BrowseName=ANN1MiningField4; name=MachiningPower; invalideValueTreatment=asIs;
usageType=predicted;
NeuralOutputsType: BrowseName=ANN1NeuralOutputs1; numberOfOutputs=1;
NeuralOutputType: BrowseName=ANN1NeuralOutput0; outputNeuron=3,0;
DerivedFieldType: BrowseName=ANN1NeuralOutput0Field; name=MachiningPower; dataType=Double; optype=continuous;
NeuralInputsType: BrowseName=ANN1NeuralInputs1; numberOfInputs=4;
NeuralInputType: BrowseName=ANN1NeuralInput0; id=0,0;
DerivedFieldType: BrowseName=ANN1NeuralInputField0; name=CuttingSpeed; dataType=Double; optype=continuous;
NeuralInputType: BrowseName=ANN1NeuralInput1; id=0,1;
DerivedFieldType: BrowseName=ANN1NeuralInputField1; name=FeedPerTooth; dataType=Double; optype=continuous;
NeuralInputType: BrowseName=ANN1NeuralInput2; id=0,2;
DerivedFieldType: BrowseName=ANN1NeuralInputField2; name=CuttingDepth; dataType=Double; optype=continuous;
NeuralInputType: BrowseName=ANN1NeuralInput3; id=0,3;
DerivedFieldType: BrowseName=ANN1NeuralInputField3; name=CuttingWidth; dataType=Double; optype=continuous;
NeuralLayerType: BrowseName=ANN1NeuralLayer0; activationFunction=logistic;
NeuronType: BrowseName=ANN1NeuralLayer0Neuron0; id=1,0; bias=-0.1503827612210782;
ConType: BrowseName=ANN1NeuralLayer0Neuron0Con0; from=0,0; weight=0.4940730222511675;
ConType: BrowseName=ANN1NeuralLayer0Neuron0Con3; from=0,3; weight=1.3754400494621 13;
ConType: BrowseName=ANN1NeuralLayer0Neuron0Con2; from=0,2; weight=1.2872593805539925;
ConType: BrowseName=ANN1NeuralLayer0Neuron0Con1; from=0,1; weight=0.6474712892546692; ...
```

Fig. 13. Example of the OPC UA PMML objects for a neural network model.

PMML documents [36]. Fig. 12 depicts a part of the PMML document for the NN-based power predictive model.

Next, Server 2 interprets and converts the PMML documents to PMML objects using JPMML SDK. Server 2 then encodes these object instances to OPC UA-compliant objects with the use of the OPC UA-compliant PMML Java classes imported from UA Code Generator. Then, Server 2 publishes these OPC UA objects.

4) Model Repository (Client 2): We also use OPC UA SDK for Client 2. Client 2 subscribes to and decodes the OPC UA objects published by Server 2. We also use OPC UA-compliant Java classes for decoding. Fig. 13 depicts a portion of the OPC UA objects used for the NN-based model. The tags ending with "Type" indicate OPC UA object types, and "BrowseName" refers to the ID name. The tags beginning with a lower case letter indicate variable types, followed by their instances. Consequently, we confirm that the OPC UA objects are coincident with the contents of the two PMML documents (i.e., the data content of Fig. 12 matches with that of Fig. 13).

D. Discussion

1) Processing Time Overhead: We measure the processing time in computer to investigate operational performance of the

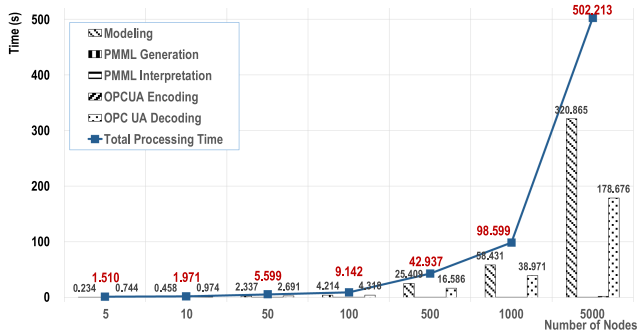

Fig. 14. Processing time for NN models.

open-source solutions used. Fig. 14 presents the processing time with regard to the increase of number of neuron nodes in NN models. The total processing time increases as the number of neurons increases. Specifically, creating NN models (tagged with "modeling") on Weka consumes the most time because operation count increases to calculate weights between neurons and bias in individual neurons. Generating and interpreting PMML objects on JPMML show low processing time of average 0.518 s (PMML generation) and 0.042 s (PMML interpretation), respectively. Transforming the PMML objects to their OPC UA objects records average 0.432 s on the server SDK (OPC UA encoding), whereas, converting the OPC UA objects to generic Java classes and variables spends much time on the client SDK (OPC UA decoding).

2) Metamodeling: OPC UA builds upon metamodeling. However, such metamodeling is generally complex [17]. It requires human's understanding of both domains (OPC UA and PMML) and the establishment of semantic integration for their mutual transfer. The merits of metamodeling stem from the construction of 1-to-N relations by providing a unified model instead of N-to-N relations that rely on separate and factorial modeling. This advantage is largely associated with the extensibility of the OPC UA, as described in Section I.

3) Tool and Programming Support: We efficiently implemented the prototype with support of open-source solutions. These solutions provide enriched functionalities, thereby reducing implementation time. Additionally, open-source solutions support other programming languages, such as C, C++, and C# to implement OPC UA integrated environments, although we used Java for this purpose. Moreover, DMG has published Portable Format for Analytics (PFA), which is a lightweight version of PMML with a JSON-based language for providing simple and flexible programming constructs and model library functions [37]. The present study developed an XML-based model interface because the current PMML only supports XML. In the future, JSON-based model interfaces can be developed to implement simple and flexible model exchange between PFA and OPC UA.

4) Interoperability With Industrial Modeling Languages and Communication Protocols: Various modeling languages exist in industries for abstracting and representing assets and processes. There are also diverse industrial communication protocols to exchange information between hierarchical and heterogeneous systems. Meanwhile, OPC UA can be a modeling language as

TABLE III
ADVANTAGES AND DISADVANTAGES

Advantages
- Enable OPC UA integration for data and model interoperability
- Provide a unique model exchange environment across industries
- Enable model transaction on 1-to-N server-client applications
- Operate independently of commercial data analysis tools
- Increase the flexibility of PMML by adding data content over the stationary PMML schema
Disadvantages
- Increase cost and time burdens due to en/de-coder implementations
- Increase complexity of model exchange due to the use of the OPC UA transport mechanism
- Necessitate the design of mapping rules and information models based on meta-modeling
- Incompatible with data analysis tools supporting PMML
- No commercial tools available for PMML-OPC UA transformation

well as a communication protocol because it provides both of information modeling and transport mechanism. In practice, OPC UA has integrated other modeling languages in collaboration with ISA-95, PackML, AutomationML (an XML-based data exchange format for transferring production engineering data), and Administration Shell (a standardized representation of industrial assets) [38], [39]. OPC UA has also released the companion specifications relevant to integrate with other communication protocols including MTConnect, PROFINIET, PowerLink, and PLCopen (a standardized interface to programming programmable logic controllers) [38]. Thus, the proposed interface can be interoperable with the other modeling languages and communication protocols that comply with the OPC UA standard. It causes from the adoption of OPC UA as a unique means. However, further implementation needs to be accompanied to integrate disparate languages and protocols with the proposed interface. In the future, we expect that a unified and standardized data and model pipeline can be embodied across manufacturing systems and CPPS when the proposed interface interconnects with PLCopen, AutomationML, and Administration Shell using OPC UA. PLCopen-OPC UA can provide field data in the vertical integration aspect; AutomationML-OPC UA do engineering data in the horizontal integration aspect; AdministrationShell-OPC UA do asset data in the value chain aspect; the proposed interface do model data in the model lifecycle aspect. This should be a future work.

5) Advantages and Disadvantages: The prototype demonstrated that PMML models could be exchanged within the OPC UA environment. Nevertheless, a question arises as to why PMML models cannot be exchanged without the OPC UA. PMML can operate alone by transferring the PMML files exported and imported by data analysis tools. However, these tools are not subordinate to but independent of manufacturing systems. The dependency of such file transaction is limited to ensure data and model interoperability simultaneously because the data and models are handled separately by manufacturing systems and data analysis tools. The present study pursues the establishment of a fully integrated OPC UA environment to ensure the interoperability of both data and models. This environment is implemented through a unified OPC UA server–client architecture and thus enables the incorporation of seamless data and model exchange. Table III summarizes the advantages and disadvantages of our method, compared with the PMML file transfer method.

VI. CONCLUSION

This article proposed an OPC UA-compliant interface for data analytics models to allow interoperable manufacturing intelligence. In the model design, the mapping rules were defined to match the PMML schema with the OPC UA model structures and protocols. The base- and domain-specific models for PMML were designed for the specifics of the regression and NN models. For the function implementation, the prototype system was designed and implemented to exchange the OPC UA-compliant PMML models in an OPC UA server–client.

The contribution of this study is the implementation of a cross-domain interface between CPPS and manufacturing systems via the exchange of OPC UA-compliant PMML models. An additional contribution is the implementation of model interoperability for the OPC UA, which currently covers data interoperability in industrial domains.

However, the present work has several limitations. Our models are neither verified nor validated with accredited organizations. The conformance and reliability of our models remains challenging. Moreover, the designed models only include regression and NN models, but the remaining 15 models (e.g., Naïve Bayes, SVM, etc.) existing in the current PMML schema are excluded. Another limitation is the use of a data simulator that only produces controlled data. Real data generators from actual machines generate not only specific but also uncontrolled, erroneous, and missing data, thus making it more difficult to create machine-specific predictive models. Future research directions include the following:

1) verification and validation of our models;
2) extension of our modeling to the remaining data analytics models;
3) implementation of a more realistic environment using actual machines and their data;
4) implementation of a fully interoperable manufacturing intelligence based on OPC UA, taking into consideration the vertical, horizontal, and value chain aspects of manufacturing systems.

REFERENCES

[1] L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Ann. Manuf. Technol.*, vol. 65, pp. 621–641, 2016.
[2] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," *Comput. Chem. Eng.*, vol. 47, pp. 145–156, 2012.
[3] C. Liu, H. Vengayil, Y. Lu, and X. Xu, "A cyber-physical machine tools platform using OPC UA and MTConnect," *J. Manuf. Syst.*, vol. 51, pp. 61–74, 2019.
[4] OPC Unified Architecture specification – Part 1: Overview and concepts, release 1.04, OPC Foundation, Scottsdale, AZ, USA, 2017.
[5] J. Woo, S. J. Shin, W. Seo, and P. Meilanitasari, "Developing a big data analytics platform for manufacturing systems: Architecture, method, and implementation," *Int. J. Adv. Manuf. Technol.*, vol. 99, pp. 2193–2217, 2018.

[6] A. Guazzelli, M. Zeller, W. C. Lin, and G. Williams, "PMML: An open standard for sharing models," *R J.*, vol. 1, no. 1, pp. 60–65, 2009.
[7] S. Ren, Y. Zhang, Y. Liu, T. Sakao, D. Huisingh, and C. M. V. B. Almeida, "A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions," *J. Cleaner Prod.*, vol. 210, pp. 1343–1365, 2019.
[8] H. Zhang, U. Roy, Y. Tsun, and T. Lee, "Enriching analytics models with domain knowledge for smart manufacturing data analysis," *Int. J. Prod. Res.*, 2019. [Online]. Available: https://doi.org/10.1080/00207543.2019.1680895
[9] O. Unver, "An ISA-95-based manufacturing intelligence system in support of lean initiatives," *Int. J. Adv. Manuf. Technol.*, vol. 65, pp. 853–866, 2013.
[10] Data Mining Group, "PMML 4.3 – General structure," Aug. 2016. [Online]. Available: http://dmg.org/pmml/v4-3/GeneralStructure.html
[11] P. O'Donovan, K. Bruton, and D. O'Sullivan, "Case study: The implementation of a data-driven industrial analytics methodology and platform for smart manufacturing," *Int. J. Progn. Health Manage.*, vol. 7026, pp. 1–22, 2016.
[12] D. Lechevalier, A. Narayanan, S. Rachuri, and S. Foufou, "A methodology for the semi-automatic generation of analytical models in manufacturing," *Comput. Ind.*, vol. 95, pp. 54–67, 2018.
[13] P. O'Donovan, C. Gallagher, K. Bruton, and D. O'Sullivan, "A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications," *Manuf. Lett.*, vol. 15, pp. 139–142, 2018.
[14] D. Lechevalier, S. Hudak, R. Ak, Y. T. Lee, and S. Foufou, "A neural network meta-model and its application for manufacturing," in *Proc. IEEE Int. Conf. Big Data*, 2015, pp. 1428–1435.
[15] J. Park *et al.*, "Gaussian Process Regression (GPR) representation in predictive model markup language (PMML)," *Smart Sustain. Manuf. Syst.*, vol. 1, no. 1, pp. 121–141, 2017.
[16] S. Nannapaneni *et al.*, "Predictive Model Markup Language (PMML) representation of Bayesian networks: An application in manufacturing," *Smart Sustain. Manuf. Syst.*, vol. 2, no. 1, pp. 87–113, 2018.
[17] W. Mahnke, S-H. Leitner, and M. Damm, *OPC Unified Architecture*. Berlin, Germany: Springer-Verlag, 2009.
[18] T. Sauter and M. Lobashov, "How to access factory floor information using internet technologies and gateways," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 699–712, Nov. 2011.
[19] P. Adolphs *et al.*, "Reference Architecture Model Industrie 4.0 (RAMI 4.0)," VDI/ZVEI, Düsseldorf, Germany, Jul. 2015.
[20] J. Schlechtendahl, M. Keinert, F. Kretschmer, A. Lechler, and A. Verl, "Making existing production systems Industry 4.0-ready: Holistic approach to the integration of existing production systems in Industry 4.0 environments," *Prod. Eng.*, vol. 9, no. 1, pp. 143–148, 2015.
[21] S. Grüner, J. Pfrommer, and F. Palm, "RESTful industrial communication with OPC UA," *IEEE Trans. Ind. Informat.*, vol. 12, no. 5, pp. 1832–1841, Oct. 2016.
[22] M. Schleipen, S. S. Gilani, T. Bischoff, and J. Pfrommer, "OPC UA & Industrie 4.0 - enabling technology with high diversity and variability," *Procedia CIRP*, vol. 57, pp. 315–320, 2016.
[23] O. Givehchi, K. Landsdorf, P. Simoens, and A. W. Colombo, "Interoperability for industrial cyber-physical systems: An approach for legacy systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3370–3378, Dec. 2017.
[24] S. Cavalieri, M. G. Salafia, and M. S. Scroppo, "Integrating OPC UA with web technologies to enhance interoperability," *Comput. Standard Interfaces*, vol. 61, pp. 45–64, 2019.
[25] S. Rohjans, M. Uslar, and H. J. Appelrath, "OPC UA and CIM: Semantics for the smart grid," in *Proc. IEEE PES Transmiss. Distrib. Conf. Expo.*, 2010, pp. 1–8.
[26] B. Lee, D. K. Kim, H. Yang, and S. Oh, "Model transformation between OPC UA and UML," *Comput. Standards Interfaces*, vol. 50, pp. 236–250, 2017.
[27] B. Katti, C. Plociennik, and M. Schweitzer, "SemOPC-UA: Introducing semantics to OPC-UA application specific methods," *IFAC PapersOnline*, vol. 51, no. 11, pp. 1230–1236, 2018.
[28] L. Bruzzone and D. F. Prieto, "An incremental-learning neural network for the classification of remote-sensing images," *Pattern Recognit. Lett.*, vol. 20, pp. 1241–1248, 1999.
[29] Prosys OPC, OPC UA Modeler. Nov. 2018. [Online]. Available: https://www.prosysopc.com/products/opc-ua-modeler/
[30] Prosys OPC, OPC UA Simulation Server. Oct. 2019. [Online]. Available: https://www.prosysopc.com/products/opc-ua-simulation-server/
[31] T. Peng and X. Xu, "Energy-efficient machining systems: A critical review," *Int. J. Adv. Manuf. Technol.*, vol. 72, pp. 1389–1406, 2014.
[32] Mitsubishi Materials, "Formulae for cutting power." [Online]. Available: http://www.mitsubishicarbide.net/contents/mhg/enuk/html/product/technical_information/information/formula4.html
[33] National Institute of Standards and Technology. Box-Behnken designs, Jan. 2003. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pri/section3/pri3362.htm
[34] Prosys OPC. OPC UA SDK for Java. [Online]. Available: https://www.prosysopc.com/products/opc-ua-java-sdk/
[35] Waikato University. Weka libraries for Java. Sep. 2018. [Online]. Available: https://github.com/Waikato/weka-3.8
[36] V. Ruusmann, Java PMML API. Mar. 2019. [Online]. Available: https://github.com/jpmml
[37] Data Mining Group, PFA: Portable Format for Analytics (version 0.8.1), Nov. 2015. [Online]. Available: http://dmg.org/pfa/index.html
[38] OPC Foundation, OPC UA Information Models. Jul. 2020. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-opc-ua-information-models
[39] Federal Ministry for Economic Affairs and Energy in Germany, Berlin, Germany, "Details of the Asset Administration Shell, Part 1 – The Exchange of Information between Partners in the Value Chain of Industrie 4.0. Version 2.0," Nov. 2019.

Seung-Jun Shin received the bachelor's degree in mechanical engineering from Korea University, Seoul, South Korea, in 2002, and the master's and Ph.D. degrees with the Department of Industrial and Management Engineering from POSTECH, Pohang, South Korea, in 2005 and 2010, respectively.

He is an Associate Professor with the Division of Interdisciplinary Industrial Studies, Hanyang University, Seoul. His current research interests include cyber-physical production systems, big data analytics in manufacturing, and environmentally-conscious design and manufacturing.