Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 9823–9830

Integrating Predictive Model Markup Language with Asset Administration Shell

Seung-Jun Shin*. Jumyung Um**

**School of Interdisciplinary Industrial Studies, Hanyang University, Seoul, Republic of Korea (Tel: 82-02-2220-2358; e-mail: sjshin@hanyang.ac.kr).*

***Department of Industrial & Management Systems Engineering, Kyung Hee University, Yongin, Republic of Korea (e-mail: jayum@khu.ac.kr)*

Abstract: The article presents a systematic approach to integrate Predictive Model Markup Language (PMML) with Asset Administration Shell (AAS) for manufacturing interoperability. The present system aims to exchange and share PMML, i.e., data analytics models, across AASs, i.e., asset representations of heterogeneous manufacturing assets. Furthermore, the present system is designed to automatically generate data analytics models on production machines, convert models into the PMML format, create AAS instances for the machines, and embed the PMML models onto the AAS instances. The article includes the design architecture, including a concept model, system architecture, information structure. An AAS client-server prototype is implemented to demonstrate the feasibility of the present system. In the prototype, a server creates and transmits the AAS that corresponds to a production machine and contains submodels associated with PMML-based energy prediction models derived by regression analysis and artificial neural network. A client receives and parses the AAS and its PMML models to predict energy consumed in the machine.

Keywords: Asset Administration Shell, Predictive Model Markup Language, Interoperability, Smart Factory, Manufacturing Intelligence, Energy Prediction

1. INTRODUCTION

Smart manufacturing is recently reaching to the implementation phase over the design phase. Accordingly, relevant reference architectures have appeared to constitute and guide physical, functional, communication, and information structures needed to implement smart manufacturing. Among these, Reference Architecture Model Industrie 4.0 (RAMI 4.0) gains attention as one of the most-popular architectures over the globe (Park et al., 2020). RAMI 4.0 was published to share a common understanding of terminologies, components, and use cases, while it emphasizes the interoperability integration to ensure compatibility between vertical and horizontal integrations. RAMI 4.0 proposes two important technologies for the interoperability integration, i.e., Open Platform Communications Unified Architecture (OPC UA) and I4.0 component. OPC UA is regarded as a feasible technology for the communication layer through secure and reliable data exchange across heterogenous devices and applications (Sierla et al., 2022). In addition, RAMI 4.0 identifies a new component, i.e., I4.0 component, which refers to a combined object that comprises a real asset and its digital model (Wagner et al., 2017).

Asset Administration Shell (AAS) corresponds to the digital model of an I4.0 component. The AAS is the standardized digital representation of an asset and the corner stone of the interoperability between the applications managing manufacturing systems (Platform Industries 4.0, 2020). AASs act as asset's information containers to represent asset's identification, communication, configuration, status, compliance, security, data, and technical functionality in a unified and standardized manner (Marcon et al., 2018). Thus, AASs facilitate the virtualization of individual manufacturing assets as well as the structurization of the manufacturing asset network that constitute vertical and horizontal integrations.

Since the first release of the AAS specification, relevant applications show an increasing trend; however, they are limited to implement manufacturing intelligence through integrating AASs with data analytics. Manufacturing intelligence represents real-time understanding, reasoning, planning and management of manufacturing processes (Davis et al., 2012). Data analytics is a process of learning or mining real data collected during operations through statistics and machine learning. Data analytics can be applied to provide insight and foresight for productivity, quality, flexibility, and energy-efficiency in manufacturing (Ren et al., 2019). In this regard, manufacturing intelligence should endow individual manufacturing assets with creating and using data analytics models (hereafter, models) to gain insight and foresight for their data-driven decision making. Furthermore, manufacturing intelligence should consider the interoperable asset network where an asset's models are exchanged and shared with other assets to embody collaborative decision making across heterogeneous devices and applications. Hence, AASs need to be integrated with models to implement asset-oriented manufacturing intelligence.

10.1016/j.ifacol.2023.10.402

Meanwhile, the data science community has recognized a unified and standardized model representation as a significant issue owing to the necessity of exchanging and sharing models across computers. Data Mining Group (DMG) released Predictive Model Markup Language (PMML) as a model representation language. The PMML is an XML-based model interchange language to represent models as well as data pre/post-processing (Guazzelli et al., 2009). The PMML is a common language and currently available in multi-purpose programming languages, such as Scala, Python, and R, and thus it is not specialized for manufacturing. An available data analysis tool has to be used by human manipulation or an encoder-and-decoder has to be implemented, when PMML is applied in manufacturing system. These requirements can incur inefficient model exchange and sharing across manufacturing assets. Hence, the PMML needs to be represented, exchanged, and shared with integrating asset information for the use in manufacturing devices and applications. This approach enables the implementation of model interoperability, which signifies the seamless exchange and sharing of models across heterogeneous manufacturing assets.

This article presents the system design architecture to integrate the PMML with the AAS for manufacturing interoperability. In the article, we propose a concept model, system architecture, and information structure to specify functional and static perspective of the present system. We implement a prototype to demonstrate the feasibility of the present system. The prototype comprises a server and client. The server creates regression and Artificial Neural Network (ANN) -based energy prediction models using real data obtained from a heat tunnel machine, converts these models into the PMML format, creates and publishes the AAS that contains PMML models as submodels. The client requests and receives the AAS to predict energy consumed in the heat tunnel.

The remainder of this article is organized as follows. Section 2 summarizes AAS and PMML. Section 3 introduces the system design, and Section 4 describes the prototype implementation. Section 5 concludes the paper.

2. AAS and PMML

Sections 2.1 and 2.2 describe the information structures and literature reviews concerning AAS and PMML, respectively.

2.1 AAS

The AAS facilitates the deployment of a unified and homogeneous data interface for manufacturing interoperability, thereby aiding in reducing to M+N data interfaces, while M*N data interfaces were necessary across diverse and heterogeneous assets in the past (Grangel-González et al., 2016).

Fig. 1 illustrates the AAS information structure, which consists of a header and body section. The header identifies a real asset and its corresponding AAS to provide an access interface among I4.0 components. The body involves a set of submodels that contain data and functions specific for an asset (Marcon et al., 2018). The data indicate digital representation, such as drawing, sensory data, manual, function blocks, and so on. The functions describe the ability to achieve a purpose and generally serve as technical functionalities (Platform Industries 4.0, 2020).

Each submodel comprises a set of submodel elements that contain various subtypes, such as a property, range, file, Binary Large Objects (Blob), operation, capability, entity, and event. The second version of the AAS specification identifies the available data formats that include XML, JSON, Resource Description Framework (RDF), Automation Markup Language, and OPC UA.

Related studies that had appeared around the first release of the AAS specification mostly involved requirements, concepts, modelling and core technologies of AASs. (Grangel-González et al., 2016) presented an RDF-based approach to model and implement AASs. (Tantik and Anderl, 2017) suggested a method to combine AASs and the object memory model to demonstrate their interoperability over the internet. (Wenger et al., 2018) connected Programmable Logic Controllers (PLCs) with their AASs for automatic device configuration. Accordingly, recent studies have shown a growing trend. They concentrate on the implementation of AAS systems or the integration of AASs with existing technologies. (Cavalieri et al., 2019) developed a method to map OPC UA information models with AAS entities. (Chilwant and Kulkarni, 2019) proposed an open framework to manage creation, ownership, maintenance, and usage of AASs in industrial systems. (Ye and Hong, 2019) suggested a common template to standardize the AAS information structure. (Motsch et al., 2021) presented an interface and submodels of electrical energy consumption on AASs in a modular skill-based production system. (Sakurada et al., 2022) developed an agent based AAS approach to enhance a digitalization process for asset intelligence and collaboration. The related studies contribute to demonstrating the usability and practicality of AASs. However, they are limited to provide a practical method to deploy models into AASs for the model interoperability environment.

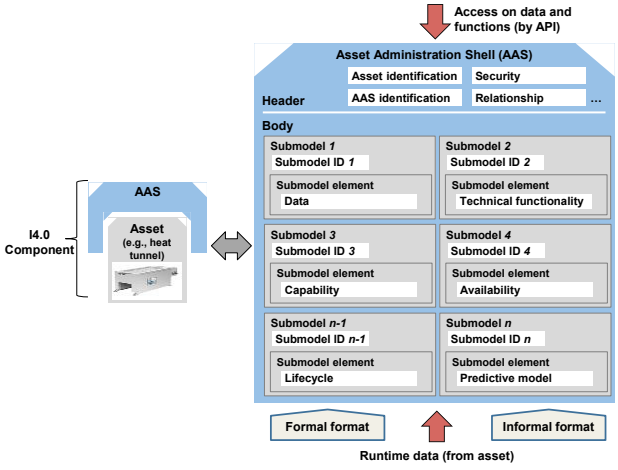

Fig. 1. AAS information structure (re-edited from (Ye and Hong, 2019))

2.2 PMML

The PMML is a computer-interpretable language for model processing, while it enables the transformation of models from mathematical equations to an XML-based structural and textual format.

Fig. 2 presents the PMML information structure, which also comprises a header and body section. The body is divided into a data dictionary, data transformation, and mining schema, target, and model specifics (Guazzelli et al, 2009). Especially, the model specifics vary in terms of model types. For example, when a model is a regression model, its model specifics include intercepts, exponents, and coefficients in the model architecture. Meanwhile, the model specifics is designated to the ANN structure when a model is an ANN model.

The PMML has been applied in manufacturing mainly for model representation and model validation. (O'Donovan et al., 2016) and (Lechevalier et al., 2018) applied PMML to represent a support vector machine model and an ANN model in air handling and milling machines, respectively. (O'Donovan et al., 2018) presented PMML-based models to be disseminated and executed by nodes in fog computing. (Park et al., 2017) designed a PMML schema for Gaussian process regression with validation of the schema in metal cutting. (Nannapaneni et al., 2018) designed and validated a PMML schema for Bayesian network in welding processes. (Shin, 2021) developed an OPC UA information model to implement compatibility between OPC UA and PMML. The related studies contribute to exhibiting the feasibility of the PMML in the manufacturing domain; however, they rarely discuss the integration of models with AASs for asset-oriented manufacturing intelligence. In addition, they did not suggest system approaches to automate the data analytics procedure, comprising data collection and pre-processing, and model creation, validation and application.

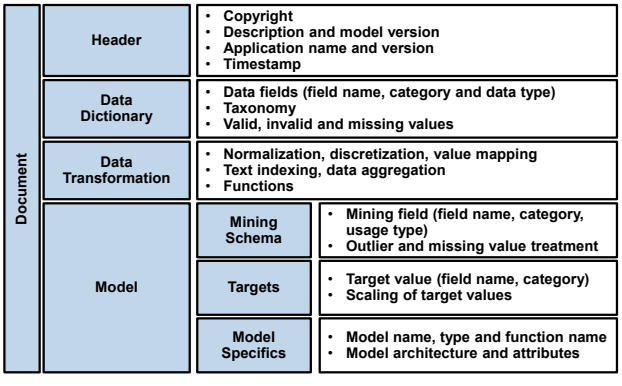

Fig. 2. PMML information structure

3. SYSTEM DESIGN

Section 3 proposes the concept model, system architecture system design and information structure to identify functional and structural perspectives of the present system. A production machine is used as an example of an asset for clear understanding.

3.1 Concept model

Assume that an asset user is eager to implement manufacturing intelligence to improve the asset's availability and productivity during the in-use phase. Essentially, an user would create models applicable for the asset. Accordingly, a technical challenge arises to concern how to create, use and exchange models on assets. It is because models were created and used by human. In addition, models were unable to be embedded to a part of an asset due to static and stationary natures of most available asset information models. Hence, it was not easy to integrate model information with asset information models because they were mutually heterogeneous. A feasible solution to integrate asset and model information is to embed model information into an asset's information container as an object-oriented approach. In this regard, the AAS is one of the most-appropriate technology for the asset representation because it provides flexible and extensible capabilities, as explained in Section 2.1. The AAS can act as a class for an asset, while it contains models with 'a-part-of' relationship. A model can be also a class and belong to a variable in the AAS class. Once a model is created, it can be subordinate to be a part of an asset. Then, the model can be used and exchanged by retrieving the model variable from the asset class. This asset-centric solution enables efficient and effective creation, use and exchange of models because an asset can contain models as its submodels. Such solution also facilitates the identification, encapsulation, and structurization of models through the relationship with an AAS as these functionalities are primary capabilities of the object-oriented approach. Meanwhile, PMML is one of the most-proper technology for model representation, as described in Section 2.2. PMML is machine-interpretable and, further, it can express models structurally and hierarchically. AASs can contain PMML models as submodels to achieve the asset-model integration. Furthermore, manufacturing intelligence can be implemented, independently of human intervention, if the asset-model integration is automated.

Fig. 3 presents the concept model. Data are collected by data interfaces while machines run in a shop floor. Predictive models can be created through training data using machine learning and they are typically represented by mathematical equations. Such models need to be converted to a machine interpretable format so that computers understand contents and semantics of the mathematical equations. PMML documents are generated based on the PMML schema to be converted from mathematically represented models. Simultaneously, an AAS instance is created with assigning its AAS identifier. This AAS instance contains PMML documents in the form of submodels with assigning their model identifiers. Once AAS instances have been created, they can be accessed with other machines and systems. Inversely, PMML documents are parsed based on the PMML schema when they are used. Mathematical equations can be extracted from PMML documents, and they can be eventually used to predict target performance.

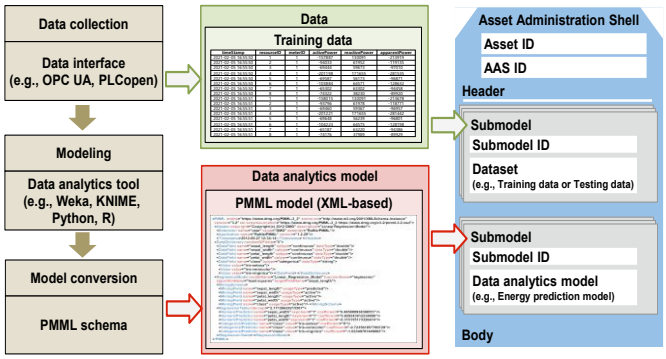

Fig. 3. Concept model

3.2 System architecture

Fig. 4 shows the system architecture to identify the structure and functions required to be implemented as a software system. This architecture is built on a server-client structure, where a server responds to the request, and a client requests and receives the relevant . The server and client can be connected based on a web service using Representational State Transfer (REST) Application Programming Interface (API). This web service enables the decoupling between a server and client. Thus, the web service can provide interoperability for system communication through fast and flexible interaction across heterogeneous systems.

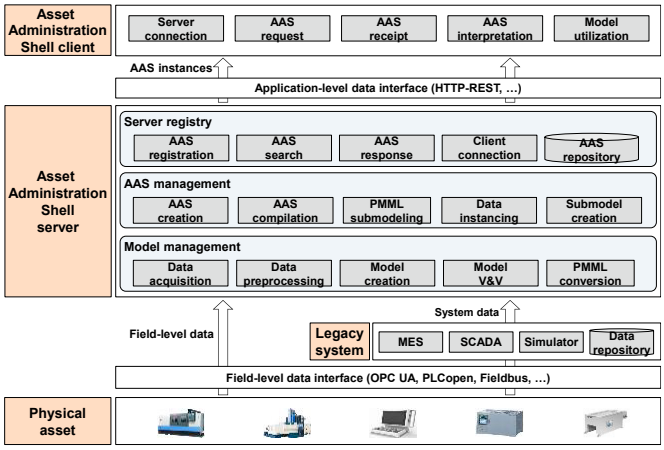

Fig. 4. System architecture

In the architecture, the server collects data from physical assets and/or legacy systems. The former provides the direct connection with assets using field-level data interfaces, such as OPC UA, and the latter provides the indirect connection with legacy systems that have processed and stored data. The server comprises *model management*, *AAS management*, and *server registry* modules. *Model management* involves: *model creation* to create models through the data analytics procedure and *PMML conversion* to convert models to PMML documents. *AAS management* includes: *AAS creation* to create AAS instances and assign their identifiers, *PMML submodeling* to embed PMML documents inside an AAS instance, and *AAS compilation* to complete generation of AAS instances. *Server registry* contains: *AAS registration* to register AAS instances, *AAS repository* to store AAS instances, and *AAS response* to transmit an AAS instance when a client requests.

Meanwhile, the client connects with the server along with complying with the security and authentication procedure. The client requests and receives AAS instances from the server. It interprets and uses the AAS instances containing PMML models for certain purposes, like energy prediction for a production machine.

3.3 Information structure

Fig. 5 depicts the information structure, which is formalized by a class diagram in Unified Modeling Language. Basically, the information structure must follow the AAS specification because this is mandatory. Instead, the AAS specification allows the variety and extensibility of data and functions by imposing various types of data elements on submodels. A submodel can be either a data type or function type, as described in Section 2.1. Here, PMML documents can be represented as both property-type and file-type data elements, while the two elements are sub-classes of the data element abstract class. If a PMML document is a property-type data element, its instances exist in a memory. If the document is a file-type data element, it exists as an XML file in a hard disk.

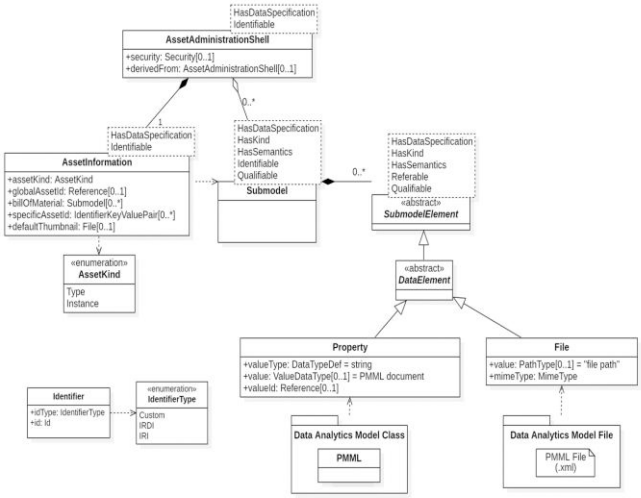

Fig. 5. Information structure

The *Identifiable*, *HasKind, HasSemantics*, *Qualifiable* and *HasDataSpecification* attributes on top of a class are independent classes to specify whether the target class: has a globally unique identifier; is either a type or instance; has a semantic definition; can be qualified; and can be extended using data specification templates, respectively (Platform Industries 4.0, 2020). The following items explain the details of main classes.

- *AssetAdministrationShell*: the top-level element associated with an AAS. It must have an identifier and correspond to a real asset. This element can contain multiple submodels.
- *AssetInformation*: the element that includes the metadata of an asset. It denotes whether an AAS belongs to a type or instance.

- *Submodel*: the element that contains digital representations and technical functionalities involved in *AssetAdministrationShell*.

- *SubmodelElement*: the abstract element suitable for description and differentiation of an asset. This element has a composition relationship with *Submodel,* and contains *DataElement,* operation, event and entity data types.

- *DataElement*: the abstract element that is inherited from *SubmodelElement* and inherits property, range, file, and Blob as sub-elements.

- *Property*: the data element that has a single value. This element must declare *valueType*, such as boolean, integer, double, float, date, duration and string, and may have *value* (a value of the property instance) and *valueId* (a unique identifier of a coded value). A PMML document can be instantiated to *value* as a string value type within this property.

- *File*: the data element that represents an address of a file. This element must define *mimeType* to state file extension and may have *value* to indicate a path and name of the referenced file. *mimeType* is formed to 'type/subtype' (e.g., application/xml). *File* provides the file path information regarding a PMML document in a hard disk or database with Multipurpose Internet Mail Extensions (MIME) types of XML file extensions. Once a client requests the retrieval of a PMML document, a server accesses to its file path and transmits the document via a file transfer.

The information structure allows the representation of an asset, data and models as primary components. Data can be included as a submodel in an AAS to keep training data and testing data for model creation and model validation, respectively. Data are sub-modelled into a file- or string property- type although the file-type is preferred to avoid memory overload. PMML models are sub-modelled, which can be represented using a string property or file- type. For example, a regression model can be represented by two submodels, which is a set of {(language, data element type) | (PMML, property-type), (PMML, file-type)}. These two disparate submodels provide a selective option for client's preference in terms of a data type.

4. PROTOTYPE IMPLEMENTATION

Section 4 explains the scenario, architecture, and result of a prototype implementation.

4.1 Implementation scenario

We use an experimental facility (FESTO Industry 4.0 Learning Factory) for a prototype implementation, as shown in Fig. 6. The facility consists of eight machines to produce simple smart phone cases. It connects with a Manufacturing Execution System (MES) to setup workplans by an operator and to conduct automated production. A PLC is attached to each machine to control the machine and collect operation data. Meanwhile, a power meter is attached to a power supply on each machine to collect power data. The operation data involve identification information regarding an assigned product, workplan, order, operation, and machine as well as start and end timestamps of an operation. The power data record a machine identifier, timestamp, and active power at 1 second interval. The operation and power data are transmitted to the MES via OPC UA on the Ethernet, and they are separately stored in two different databases.

Fig. 7 illustrates an implementation scenario. Among the eight machines, the heat tunnel machine is chosen as a target physical asset. In the scenario, an AAS server creates regression and ANN -based energy prediction models by learning data, and transmits an AAS instance that embeds PMML models as property and file data elements. Meanwhile, a client receives the AAS instance, interprets the two types of PMML models, while it acts as a Factory Energy Management System (FEMS). The client uses the models to predict the energy consumed in the heat tunnel by inputting process parameters, i.e., heating temperature and heating time.

Fig. 6. Experimental facility

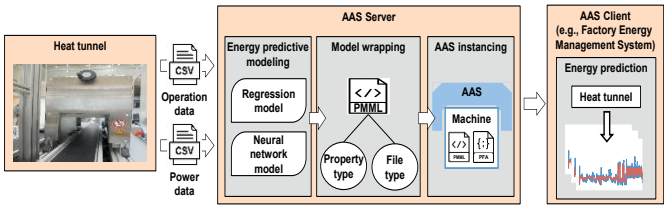

Fig. 7. Implementation scenario

Energy can vary depending on heating temperature and heating time because the former and the latter may affect active power and heating duration, respectively. Energy (E) is calculated from the integral of power (P) over time (t), or the mensuration-by-parts, as expressed in (1). E can be predicted by a statistical model or a machine learning model that are derived from training data. Equation (2) expresses a multiple linear regression-based energy prediction model. Equation (3) expresses an ANN-based energy prediction model, where an output neuron is calculated by a weighted summation over the output of hidden neurons to the process parameters.

$$E = \int_{t=0}^{T} P(t)dt = \sum_{i=0}^{M} P_i t_c \quad (1)$$

where, $P(t)$ and P_i: a power value at a timestamp, T: an operation time, M: the number of power values, t_c: the sensing time interval

$$E_{pred} = a_0 + a_1 x_1 + a_2 x_2 + \varepsilon \quad (2)$$

where, E_{pred}: a predicted energy, x: a process parameter, a_0: an intercept, a_k: a coefficient, ε: an error.

$$o_i(x_k) = \sum_{j=1}^{L} w_{ij}\theta_j(x_k) + w_{bias,i} \quad (3)$$

where, o_i: an output neuron, w_{ij}: a weight, θ_j: an activation function, L: the number of neurons, $w_{bias,i}$: a bias.

4.2 Implementation architecture

Fig. 8 presents the implementation architecture for a protype. Table 1 lists the tools used for the implementation. This architecture is derived based on the system architecture presented in Fig. 4. The prototype is implemented on the Java environment. In the prototype, we endeavour to automate all functions in the server and client except the data acquisition. The data acquisition is unavoidably manually worked because of the unavailability of direct data access by the security firewall. The validity of PMML documents is confirmed using KNIME Analytics by checking whether PMML documents are opened and interpreted well in the PMML reader function.

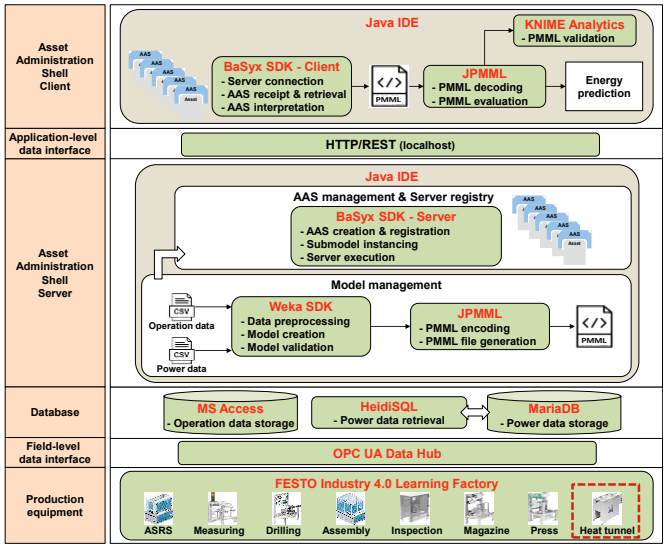

Fig. 8. Implementation architecture

Table 1. Details of tools

Tool	Use in prototype
OPC UA Data Hub	A unified data interface to transmit operation data from PLCs and power data from power meters to MES and databases
Microsoft Access	A database to store, manage and retrieve operation data
MariaDB	A database to store and manage power data
HeidiSQL (9.4.0)	A query editor to retrieve power data by access to MariaDB
Eclipse IDE (4.22.0)	The Java-based programming platform to implement server and client applications
BaSyx Java SDK (1.0.2)	A Software Development Kit (SDK) to implement an AAS server and client
Weka SDK (3.8.6)	An SDK to pre-process data and to create regression and neural network models
JPMML (1.6.3)	A library to generate PMML documents in a server and to parse PMML documents in a client
KNIME Analytics (4.5.2)	A stand-alone software to validate conformance of PMML documents

4.3 Implementation result

(a) Server

We apply Central Composite Design (CCD) as Design of Experiments for data generation. We gather operation and power data under the process parameters designated by the CCD (given by the two process parameters in the training type of Table 2). The sever starts to run once the two data are inputted to the server. Table2 lists a dataset used to create energy prediction models. Data integration is conducted to combine raw operation and power data into a dataset. This can be achieved by using start and end timestamps of an operation from the operation data. The power values in-between the start and end timestamps become an actual net energy value because they are consumed to conduct to the operation. Data normalization is performed to align minimum-to-maximum ranges with an identical zero-to-one scale at the three individual attributes.

Table 2. Dataset

Type	Heating temperature (℃)	Heating time (s)	Actual energy (J)
Training	55.6	71.2	60719.2
	45.0	20.0	4533.8
	45.0	50.0	12711.1
	45.0	80.0	39409.0
	45.0	50.0	11895.3
	60.0	50.0	42900.0
	34.4	71.2	11910.9
	45.0	50.0	12046.1
	34.4	28.8	3720.2
	45.0	50.0	12016.1
	45.0	50.0	11726.9
	55.6	28.8	14507.6
	30.0	50.0	4662.9
Testing	30.0	50.0	4739.3
	37.4	65.0	12185.9
	45.0	80.0	39053.2
	52.5	65.0	38681.7
	60.0	50.0	70714.7
	52.5	35.0	10347.9
	45.0	20.0	4517.2
	37.5	35.0	4655.8

The server creates regression and ANN models through learning the training dataset and measures the model performances using the testing dataset. The hyperparameter of the regression model is greedy as the attribution selection method; the hyperparameter parameters of the ANN model are 1 hidden layer, 4 neurons in the hidden layer, logistic activation function, 0.3 learning rate, 0.2 momentum, and 500 epochs. Equation (4) expresses an regression-based energy prediction model. This regression model obtains 0.121 Mean Absolute Error (MAE), 76.4% R^2-adjusted, and 43.5% Relative Absolute Error (RAE). Meanwhile, the ANN model gains 0.110 MAE, 0.181 RMSE, and 39.4% RAE. Note that marginal model performances appear due to low relevance between the two process parameters and the energy; however, a discussion regarding the model performance is out-of-the scope of this article.

$$\hat{E} = 0.600 Heating\ Temperature + 0.548 Heating\ Time - 0.351 \quad (4)$$

The server converts the regression and ANN models individually to PMML documents based on the PMML format. Fig. 9 shows the PMML document of the regression model (note that only the regression model will be shown due to the page limitation).

The server creates an AAS instance with assigning its identifier. The server subsequently instantiates the two PMML documents as submodels on the AAS instance. Fig. 10 presents the AAS instance, represented by YAML (a format for easy-to-read JSON). The AAS instance involves the four submodels that are composed of a combinatorial set of a regression or ANN model and a property or file type. Fig. 11 shows the submodel regarding the regression model with a string property-type, and Fig. 12 presents the submodel regarding the regression model with a file-type.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PMML xmlns="http://www.dmg.org/PMML-4_2" xmlns:data="http://jpmml.org/jpmml-model/InlineTable" version="4.2">
  <Header copyright="aml.hanyang.ac.kr" description="PMMLRegressionModelbyWeka">
    <Extension name="SSJ"/>
    <Application name="JPMML" version="1.4.9"/>
  </Header>
  <DataDictionary numberOfFields="3">
    <DataField name="HeatingTemperature" optype="continuous" dataType="double">
      <Interval closure="closedClosed" leftMargin="0.0" rightMargin="1.0"/>
    </DataField>
    <DataField name="HeatingTime" optype="continuous" dataType="double">
      <Interval closure="closedClosed" leftMargin="0.0" rightMargin="1.0"/>
    </DataField>
    <DataField name="Energy" optype="continuous" dataType="double">
      <Interval closure="closedClosed" leftMargin="0.0" rightMargin="1.0"/>
    </DataField>
  </DataDictionary>
  <RegressionModel modelName="LinearRegression" functionName="regression"
   algorithmName="LinearRegression" modelType="linearRegression" targetFieldName="Energy"
   normalizationMethod="none">
    <MiningSchema>
      <MiningField name="HeatingTemperature" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="HeatingTime" usageType="active" invalidValueTreatment="asIs"/>
      <MiningField name="Energy" usageType="predicted" invalidValueTreatment="asIs"/>
    </MiningSchema>
    <Output>
      <OutputField name="Energy" optype="continuous" dataType="double" feature="predictedValue" value="Energy"/>
    </Output>
    <RegressionTable intercept="-0.351" targetCategory="Energy">
      <NumericPredictor name="HeatingTemperature" exponent="1" coefficient="0.6"/>
      <NumericPredictor name="HeatingTime" exponent="1" coefficient="0.548"/>
    </RegressionTable>
  </RegressionModel>
</PMML>
```

Fig. 9. PMML document of regression model

```
conceptDictionary: []
identification:
 idType: Custom
 id: eclipse.basyx.aas.heattunnel
idShort: heattunnel
dataSpecification: []
modelType:
 name: AssetAdministrationShell
asset:
 identification:
  idType: Custom
  id: eclipse.basyx.asset.heattunnel
 idShort: heattunnelasset
 kind: Instance
 dataSpecification: []
 modelType:
  name: Asset
 embeddedDataSpecifications: []
embeddedDataSpecifications: []
views: []
submodels:
- keys:
  - idType: Custom
    type: AssetAdministrationShell
    value: eclipse.basyx.aas.heattunnel
    local: true
  - idType: Custom
    type: Submodel
    value: eclipse.basyx.submodel.pmml.regression
    local: true
- keys:
  - idType: Custom
    type: AssetAdministrationShell
    value: eclipse.basyx.aas.heattunnel
    local: true
  - idType: Custom
    type: Submodel
    value: eclipse.basyx.submodel.pmml.neuralnetwork
    local: true
- keys:
  - idType: Custom
    type: AssetAdministrationShell
    value: eclipse.basyx.aas.heattunnel
    local: true
  - idType: Custom
    type: Submodel
    value: eclipse.basyx.submodel.pmml.regression.file
    local: true
- keys:
  - idType: Custom
    type: AssetAdministrationShell
    value: eclipse.basyx.aas.heattunnel
    local: true
  - idType: Custom
    type: Submodel
    value: eclipse.basyx.submodel.pmml.neuralnetwork.file
    local: true
```

Fig. 10. AAS instance of heat tunnel

(b) Client

The client requests the AAS instance using its identifier and receives it from the server. The client can interpret the AAS instance and parses the submodels associated with the PMML documents. The client can obtain the mathematical equations, such as Equation (4), because a PMML document provides the model specifics that include intercepts, exponents, and coefficients in the regression model architecture. Lastly, the client outputs a predicted energy value when an operator inputs a heating time and heating time. For example, the client predicts the consumption of 37597.1 J energy when 50 °C of a heating temperature and 70 sec of a heating time are inputted by an operator.

```
parent:
 keys:
 - idType: Custom
   type: AssetAdministrationShell
   value: eclipse.basyx.aas.heattunnel
   local: true
identification:
 idType: Custom
 id: eclipse.basyx.submodel.pmml.regression
idShort: pmmlregressionpropertysubmodel
kind: Instance
dataSpecification: []
modelType:
 name: Submodel
embeddedDataSpecifications: []
submodelElements:
- parent:
   keys:
   - idType: Custom
     type: Submodel
     value: eclipse.basyx.submodel.pmml.regression
     local: true
 idShort: pmmlregressionproperty
 kind: Instance
 valueType: string
 modelType:
  name: Property
 value: <?xml version="1.0" encoding="UTF-8" standalone="yes"?><PMML version="4.2"
  xmlns="http://www.dmg.org/PMML-4_4" xmlns:data="http://jpmml.org/jpmml-model/InlineTable"><Header
  copyright="aml.hanyang.ac.kr" description="PMMLRegressionModelbyWeka">
  ... (see the content in Figure 9)
  <RegressionTable intercept="-0.351" targetCategory="Energy"><NumericPredictor name="HeatingTemperature"
  exponent="1" coefficient="0.6"/><NumericPredictor name="HeatingTime" exponent="1"
  coefficient="0.548"/></RegressionTable></RegressionModel></PMML>
```

Fig. 11. Property-type submodel of regression model

```
parent:
 keys:
 - idType: Custom
   type: AssetAdministrationShell
   value: eclipse.basyx.aas.heattunnel
   local: true
identification:
 idType: Custom
 id: eclipse.basyx.submodel.pmml.regression.file
idShort: pmmlregressionfilesubmodel
kind: Instance
dataSpecification: []
modelType:
 name: Submodel
embeddedDataSpecifications: []
submodelElements:
- parent:
   keys:
   - idType: Custom
     type: Submodel
     value: eclipse.basyx.submodel.pmml.regression.file
     local: true
 idShort: pmmlregressionfile
 kind: Instance
 modelType:
  name: File
 mimeType: application/pmml
 value: ".. \\data\\output\\8_LinearRegressionModel.pmml"
```

Fig. 12. File-type submodel of regression model

5. CONCLUSION

This article presented the design and implementation of integrating the PMML with AASs for interoperable manufacturing intelligence. This study contributes to integrate the two heterogeneous standards originated from the data science and manufacturing domains. The AAS becomes an essential technology for vertical and horizontal integrations. In this regard, this study demonstrated the feasibility and usability of AASs particularly for AAS-driven model interoperability. This study also contributes to automate data analytics and AAS modelling in a server-client architecture.

However, the prototype was implemented for a single machine. This limited implementation lacks in demonstrating practicability in a full-scale shop floor. The prototype did not include the automation of data acquisition due to the unavailability of direct data access. Data acquisition needs to be fully automated because it is time consuming and labour

intensive. This study only embodied the PMML into the AAS. Because AASs can serve versatility and extensibility as an information container, more languages and formats need to be integrated with the AAS environment to improve feasibility and practicability of AASs.

ACKNOWLEDGEMENT

This research was supported by the Ministry of SMEs and Startups, Republic of Korea, under 'Continuous Process Manufacturing Standardization of Shared Data between Facilities/Factories/Businesses in Characteristic Industries' in 'Smart Manufacturing Innovation R&D Program' (RS-2022-00140694). This work was also supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-00155911, Artificial Intelligence Convergence Innovation Human Resources Development (Kyung Hee University)).

REFERENCES

Cavalieri, S., Mulé, S., Salafia, M.G. (2019) OPC UA-based Asset Administration Shell. *45th Annual Conference of the IEEE Industrial Electronics Society*, 2982-2989, Lisbon, Portugal.

Chilwant, N., Kulkarni, M.S. (2019) Open Asset Administration Shell for industrial systems. *Manufacturing Letters*, 20, 15–21.

Davis, J., Edgar, T., Porter, J., Bernaden, J., Sarli, M. (2012) Smart manufacturing, manufacturing intelligence and demand-dynamic performance. *Computers and Chemical Engineering*, 47, 145– 156.

Grangel-González, I., Halilaj, L., Coskun, G., Auer, S., Collarana, D., Hoffmeister, M. (2016) Towards a semantic administrative shell for Industry 4.0 components. *IEEE 10th International Conference on Semantic Computing*, 230-237, Laguna Hills, U.S.A.

Guazzelli, A., Zeller, M., Lin, W.C., Williams., G. (2009) PMML: An open standard for sharing models. *The R Journal*, 1(2), 60-65.

Lechevalier, D., Narayanan, A., Rachuri, S., Foufou, S. (2018) A methodology for the semi-automatic generation of analytical models in manufacturing. *Computers in Industry*, 95, 54–67.

Marcon, P., Diedrich, C., Zezulka, F., Schröder, T., Belyaev, A., Arm, J., Benesl, T., Bradac, Z., Vesely., I. (2018) The asset administration shell of operator in the platform of Industry 4.0. *18th International Conference on Mechatronics*, Brno, Czech Republic.

Motsch, W., Sidorenko, A., David, A., Rübel, P., Wagner, A., Ruskowski, M. (2021) Electrical energy consumption interface in modular skill-based production systems with the Asset Administration Shell. *Procedia Manufacturing*, 55, 535–542.

Nannapaneni, S., Narayanan, A., Ak, R., Lechevalier, D., Sexton, T., Mahadevan, S., Lee, Y.T.T. (2018) Predictive Model Markup Language (PMML) representation of Bayesian networks: An application in manufacturing. *Smart and Sustainable Manufacturing Systems*, 2(1), 87-113.

O'Donovan, P., Bruton, K., O'Sullivan, D.T. (2016) Case study: the implementation of a data-driven industrial analytics methodology and platform for smart manufacturing. *International Journal of Prognostics and Health Management*, 7026.

O'Donovan, P., Gallagher, C., Bruton, K., O'Sullivan, D.T.J. (2018) A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications. *Manufacturing Letters*, 15, 139–142.

Park, J., Lechevalier, D., Ak, R., Ferguson, M., Law, K.H., Lee, Y.T.T., Rachuri, S. (2017) Gaussian process regression representation in Predictive Model Markup Language (PMML). *Smart and Sustainable Manufacturing Systems*, 1(1), 121–141.

Park, K.T., Yang, J., Noh, S.D. (2020). VREDI: Virtual representation for a digital twin application in a work-center-level asset administration shell. *Journal of Intelligent Manufacturing*, 32, 501-544.

Platform Industries 4.0. (2020) Details of the Asset Administration Shell: Part 1 - The exchange of information between partners in the value chain of Industrie 4.0. Specification, version 3.0RC01.

Ren, S., Zhang, Y., Liu, Y., Sakao, T., Huisingh, D., Almeida., C.M.V.B. (2019) A comprehensive review of big data analytics throughout product lifecycle to support sustainable smart manufacturing: A framework, challenges and future research directions. *Journal of Cleaner Production*, 210, 1343-1365.

Sakurada, L., Leitao, P., De la Prieta, F. (2022) Agent-based asset administration shell approach for digitizing industrial assets. *IFAC PapersOnLine*, 55(2), 193–198.

Shin, S.J. (2021) An OPC UA-compliant interface of data analytics models for interoperable manufacturing intelligence. *IEEE Transactions on Industrial Informatics*, 17(5), 3588-3598.

Sierla, S., Azangoo, M., Rainio, K., Papakonstantinou, N., Fay, A., Honkamaa, P., Vyatkin, V. (2022) Roadmap to semi-automatic generation of digital twins for brownfield process plants. *Journal of Industrial Information Integration*, 27, 1050282.

Tantik, E., Anderl, R. (2017) Integrated data model and structure for the asset administration shell in Industrie 4.0. *Procedia CIRP*, 60, 86–91.

Wagner, C., Grothoff, J., Epple, U., Drath, R., Malakuti, S., Grüner, S., Hoffmeister, M., Zimermann, P. (2017) The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. *22nd IEEE International Conference on Emerging Technologies and Factory Automation*, Limassol, Cyprus.

Wenger, M., Zoitl, A., Müller, T. (2018) Connecting PLCs with their Asset Administration Shell for automatic device configuration. IEEE 16th International Conference on Industrial Informatics, 74-79, Porto, Portugal.

Ye, X., Hong, S.H. (2019) Toward Industry 4.0 components: Insights into and implementation of Asset Administration Shells. *IEEE Industrial Electronics Magazine*, 13(1), 13-25.