Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Fuzzy model xml formulation for production dynamics analysis and control

Gorazd Karer * **Igor Škrjanc** * **Gašper Mušič** *

* *Faculty of Electrical Engineering, University of Ljubljana, Slovenia (e-mail: gorazd.karer@fe.uni-lj.si).*

Abstract: The paper introduces an XML schema complying with PMML standard suitable for a general fuzzy model formulation. It also presents a graphical user interface that is a part of a tool for analysis and control of production dynamics, which was developed at the Competence Center for Advanced Control Technologies. The interface is used for *eFuMo* model identification, validation and simple conversion from an *eFuMo* model to a standardized XML message and vice versa. The conversion can be carried out via a DOM object, which enables manipulation of XML structure in Matlab, or directly. The XML files that comply with the PMML standard provide a way for applications to describe and exchange models produced by data-mining and machine-learning algorithms. The goal of the presented work is to provide a platform for including fuzzy models into the PMML framework and thus stimulate interoperability of various applications that take advantage of fuzzy models, e.g. in production dynamics analysis and control, as well as in other implementations.

1. INTRODUCTION

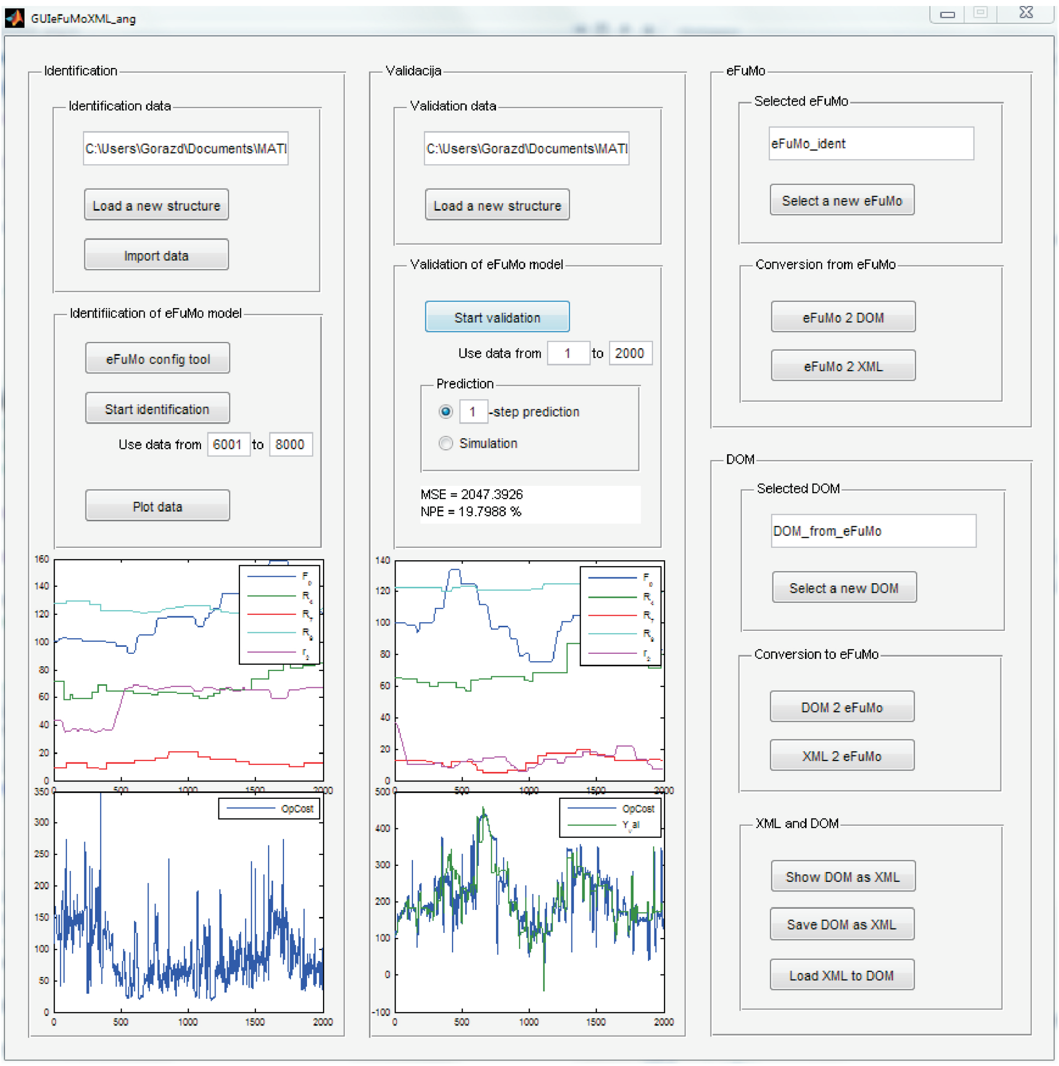
The production management is typically divided to several hierarchical levels, according to IEC 62264 standard (see IEC [2007]). The topmost level involves economic analysis and planning of future strategic goals. The goals are then presented to the lower (production) level in order to schedule and divide the production tasks. On this level, both strategic goals and dynamical properties and limitation of the production process have to be taken into account. Usually, the production process optimization is carried out on this level as well. The lowest level deals with the production process control, where suitable control loops ensure stable operation within process constraints.

Due to a large number of variables that influence production performance, effective optimization and control on the production level is generally a very complex task, which is usually carried out by production managers. However, the lack of tools for decision-making support often results in non-optimal decisions.

In the increasingly global competition on product quality and production cost, many manufacturing companies adjust their production by focusing on customized product and fast time to market. Modular simulation and modeling enable a flexible production and rapid product innovation. *Cyber-Phisical-Systems* within a factory of the future will need to enable communication between humans, machines and products (see Brettel et al. [2014]). However, suitable production models are needed to achieve this goal. On the production level, several variations of MES (Manufacturing Execution Systems) can be applied. However, MES support is frequently limited to archiving and displaying production variable measurements and lacks solid support for production optimization and control. Therefore, the concept of holistic production control has been introduced by Zorzut et al. [2009]. Holistic production control employs production data analysis in order to obtain suitable models of production dynamics that allow for effective optimization of production with regard to the strategic goals. The idea is to have a decision-making support system that would help the production manager to make effective decisions so as to optimize the production process (see Glavan et al. [2013b,a]).

The Competence Center for Advanced Control Technologies (see CCACT [2013]) is a Slovenia-based research-development center focusing on new developments in control technology (automation, computerization and cyberneticisation). Within the Center, a tool for production dynamics analysis and control has been developed.The tool has a modular design, which allows for a clear and comprehensible structure, distributed operation on several systems and possibility of a later development and addition of new modules that are not limited to one programming language. In order to successfully integrate the tool into a production process, it is vital to provide support for a standardized form of messages that are used for communication in business and production information systems. Therefore, XML messages have been chosen for this task. Their structure can be generally defined in a xsd schema, as described in the PMML (Predictive Model Markup Language) standard (see DMG [2013]). The goal of the PMML standard is to provide a unified formulation of a wide range of classes of data-mining models and hence enable interoperability of various applications and tools from different manufacturers by simplifying the model and data transfers among various environments.

In the paper, we propose an XML structure that supports a general fuzzy model formulation and complies with the PMML standard. Furthermore, we present a part of the tool for production analysis and control that provides a graphical user interface with user-friendly functions for evolving identification of fuzzy models and functions for



C:\Users\Gorazd\Documents\MATI

Use data from 6001 to 8000

C:\Users\Gorazd\Documents\MATI

Use data from 1 to 2000

(●) 1 -step prediction

MSE = 2047.3926
NPE = 19.7988 %

eFuMo_ident

DOM_from_eFuMo

Fig. 1. Graphical user interface

their conversion to a standardized XML message and vice versa.

2. FUZZY MODEL FORMULATION BASED ON XML

Among other complex dynamical properties many real industrial processes express distinct nonlinearities. A well established approach to modeling nonlinear systems nonlinearities employs *fuzzy logic*, see Babuška [1996], Babuška and Verbruggen [1996], Škrjanc and Matko [2001], Škrjanc et al. [2005].

Fuzzy logic emerges from the theory of *fuzzy sets* (see Zadeh [1965]), which introduces the concept of the *degree of membership* into the classic theory of *crisp sets* (see Babuška [1998]). An element is not just *included* or *not included* in a particular set, but can also be a *fuzzy member* of that set, i.e., can be included to a certain degree between 0 (*not included*) and 1 (*included*).

In this manner it is possible to include some adverbs expressing a degree or extent, such as *very, rather, moderately* etc. in the theory of sets. Fuzzy sets also introduce some linguistic variables, which can have some linguistic value that can be interpreted using fuzzy sets and membership degrees.

An advantage of the fuzzy models is the intuitive description of causal dynamical connections between inputs and outputs of a system using *if–then* sentences (see Girimonte and Babuška [2004]). An essential property of the fuzzy model is that they can be regarded as universal approximator of nonlinear dynamics (see Castro [1995], Girosi and Poggio [1990]). It is possible to approximate any continuous nonlinear function to an arbitrary precision using fuzzy models.

Fuzzy logic and fuzzy models represent a useful derivative of the theory of fuzzy sets and can be employed in modeling of real complex dynamical systems.

Fuzzy models can be formulated as XML messages, but no general standard has been widely accepted yet. There are several XML solutions that are mainly intended for fuzzy controller formulation, such as FCL (see IEC [2013]) with IEC 61131-7 standard, FML (see Acampora [2013]). However, these formulations are not adequate for a general fuzzy models, such as the models used for production dynamics analysis and control.

2.1 PMML standard

There is an evident need for a standardized formulation of fuzzy models. Therefore, we developed a XML formulation that is based on the PMML standard (see DMG [2013]), which was slightly adapted so as to include fuzzy model formulations. The basic PMML schema comprises several various elements that can be used to describe a model. There is also a set of useful data structures defined in the schema.

According to the PMML schema, the root element of the XML file containing the model is called <PMML>. The element <PMML> includes an element <Header>, which contains the header of the XML file, and an element <DataDictionry>, which contains the description of the data used in the model. In general, <PMML> can also include an element from the <MODEL-ELEMENT> group, which is described later. Furthermore, it can include a <MiningBuildTask> element, which describes the learning (identification) data, and a <TransformationDictionary> element comprising the eventual data transformations, such as normalization, discretizing etc.

The group <MODEL-ELEMENT> supports the following model formulations: AssociationModel, BaselineModel, ClusteringModel, GeneralRegressionModel, MiningModel, NaiveBayesModel, NearestNeighborModel, NeuralNetwork, RegressionModel, RuleSetModel, SequenceModel, Scorecard, SupportVectorMachineModel, TextModel, TimeSeriesModel, and TreeModel. Despite the large set of possible models it does not comprise a formulation that is suitable for a general fuzzy model. Therefore, we used the extension element, which is supported in the PMML schema: instead of a model from the <MODEL-ELEMENT> group[1], we used a ne subelement called <Extension>. <Extension> allows for user-defined contents and can hence be used to store the whole structure enabling the reconstruction of a fuzzy model.

2.2 Structure of the XML formulation for fuzzy model within the PMML standard

The XML formulation must meet three main requirements:

- It complies with the PMML standard, which means that it can be validated against the XSD Schema in file pmml-4-1.xsd (can be downloaded from DMG [2013]).
- It supports a most general fuzzy model formulation.
- It has a clear and comprehensible structure so that it is possible to derive the fuzzy model from the XML file without any particular tools or additional knowledge.

The basic structure of XML file is defined in the PMML standard. As described later, we have used the elements and data structures provided by the PMML standard for the fuzzy model formulation ta a large extent. We have also taken to the rules defined in the PMML into consideration.

the first compulsory subelement of the root element <PMML> is <Header>. It can have two attributes: *copyright* and *description.* In our case *copyright* can contain the author of the XML formulation. The <Header> element supports several subelements: <Application> element, <Annotation> element and <Timestamp> element. In our case, the <Annotation> element, can provide a comment on the fuzzy model, whereas the <Timestamp> element incorporates the time of creation of the XML formulation.

[1] A model from the <MODEL-ELEMENT> group is not compulsory in the PMML schema.

The next compulsory subelement of the root element <PMML> is <DataDictionary>, which can have a nonnegative integer attribute *numberOfFields.* <DataDictionary> element also comprises a subelement <DataField >, which can have the following attributes: *name*, which must be of a FIELD-NAME type; *optype*, which can be either *categorical*, *ordinal* or *continuous*; *dataType* with a limited set of types; *displayName*; *taxonomy*; and *isCyclic.* The attributes *name*, *optype* and *dataType* are compulsory, whereas the attributes *displayName*, *taxonomy* and *isCyclic* are not.

The next subelement of the root element <PMML> is usually from the <MODEL-ELEMENT> group. As mentioned, the group does not comprise a formulation that would be suitable for a general fuzzy model. Therefore, we used the <Extension> element, which is supported in the PMML schema and can be used to store all the data needed for the reconstruction of the fuzzy model. The <Extension> element can have three attributes: *extender*, *name* and *value.* In our case we use the attribute *name* that contains the string "fuzzyModel", denoting that the element contains a fuzzy model. The <Extension> element comprises a subelement <FuzzyModel>, which supports a general fuzzy model. The element can have several attributes: *functionName*, defining a function, *modelName* with the name the fuzzy model and *modelType* containing the type of the model. The <FuzzyModel> element comprises five subelements: <MiningSchema>, which references the appropriate signals used in the model; <RegressorDefinition>, which defines the signals and the delays making up the regressor; <OutputDefinition>, which defines the output signals of the model, with their respective delays; <Rules>, containing the rule-set of the fuzzy model; and <GeneralParameters>, which comprises the other parameters that are crucial for model reconstruction.

The <RegressorDefinition> element has an attribute *numberOfInputs*, with the number of particular signals making up the regressor of the model. Each signal in the regressor result in a <RegressorInput> subelement, with its identification attribute *id* and a user-friendly name, which describes the particular regressor signal and its delay, e.g. $F_p(k-4)$. The <RegressorInput> element contains a <Delay> subelement, defining the delay, and a <DerivedField> complex subelement, which is defined in the PMML standard and references the appropriate signal from the <DataDictionary> set.

In a similar way as the <RegressorDefinition> element, the <OutputDefinition> element defines the output signals in its <FuzzyModelOutput> subelements.

The <Rules> element has a *numberOfRules* attribute, which contains the number of rules in the fuzzy model. The element comprises an appropriate number of <Rule_i> subelements, each with its identification attribute *i.* Each <Rule_i> element is made up of the antecedent part, defined in the <IfPart> subelement, and the consequent part, defined in the <ThenPart> subelement.

The antecedent part comprises the fuzzy membership function in the <MembershipFunctionDefinition> subelement. The subelement has a *type* attribute, which in our case contains the string c̈luster·. This is because the membership functions in the *eFuMo* model are defined as clusters. However, in general, any other type of membership functions could be used instead.

A cluster defining a membership function is formulated as a standard PMML element <Cluster>. In our case, it comprises two standard PMML subelements: <Array> element defines the center of the cluster, whereas <Covariances> element defines the shape of the cluster using a covariance matrix.

The consequent part <ThenPart> has a *type* attribute, which in our case contains the string l̈ocalLinearModel·. It contains a <LocalLinearModel> subelement, where the local linear model appurtenant to the respective rule in <Rule_i> is defined.

The <GeneralParameters> element comprises a <NumberOfOutputs> subelement, which contains the number of outputs, and the parameters defining the method for calculating the membership values contained in the <MembershipDegreeCalculationParameters> subelement. The latter is made up of <MembershipDegreeCalculatingMethod>, <FuzzinessFactor> and <OverlappingFactor> subelements.

Several elements comprise data formulated in a matrix form. Therefore, in such a case the standard PMML element <Matrix> is used. The element is made up of one subelement <Array> for each row in the matrix. The <Array> element is also predefined by the PMML standard. In our case, the compulsory *type* attribute contains the string r̈eal·. The <Array> element contains the element values in the particular row of the matrix separated by spaces.

Below there is a collapsed example of an XML file containing a fuzzy model.

```
<?xml version="1.0" encoding="utf-8"?>
<PMML>
   <Header .../>
   <DataDictionary .../>
   <Extension>
      <FuzzyModel>
         <MiningSchema .../>
         <RegressorDefinition .../>
         <OutputDefinition .../>
         <Rules .../>
         <GeneralParameters .../>
      </FuzzyModel>
   </Extension>
</PMML>
```

2.3 eFuMo fuzzy model

In our case, production dynamics has been modeled using a powerful Matlab tool for evolving identification of fuzzy models *eFuMo (evolving fuzzy model)*, which was developed based on the algorithms published in Dovžan and Škrjanc [2010a,b, 2011]. The method for identifying dynamical systems includes recursive input-output space partitioning using several clustering methods. Furthermore, it enables identification of local linear models within their respective clusters. In addition, it comprises the algorithms for effectively increasing and decreasing the number of clusters during the identification procedure. The result of the identification is an *efm* type Matlab object. The object includes the structure and all parameters of the identified fuzzy model. What is more, it comprises all the settings used in the identification and evolving clustering algorithms. However, all this data makes the *efm* object itself a rather extensive and complex structure. On the other hand, when using a fuzzy model for predicting the behavior of the modeled system all these setting are not needed. Therefore, we are can use only a limited amount of data of the identified model in order to be able to fully reconstruct the fuzzy model for prediction.

Let us assume we have an eFuMo object named named `eFuMo` loaded in the Matlab workspace. The set of data that allows to fully reconstruct the fuzzy model for prediction is made up of the following parts of the eFuMo object: the distance definition `eFuMo.id_md_calc_method`; the number of identified clusters that can be obtained using `n_clust=size(eFuMo.clust,2)`; the number by which the problem-space dimension decreases when projecting from the input-output into the input space `eFuMo.projekcija`; a selected set of data on the identified clusters that are found under `eFuMo.clust`.

As mentioned, each cluster is defined using a selected set of data that are found under `eFuMo.clust`. The sets are made up of: cluster index `i`; a matrix with the center coordinates of i-th cluster `eFuMo.clust(i).Vi`; a fuzzy covariance matrix defining the shape of i-th cluster `eFuMo.clust(i).Fi_input`; parameters of the local linear model appurtenant to i-th cluster `eFuMo.clust(i).teta`; fuzziness parameter of fuzzy sets `eFuMo.clust(i).n`; and distance definition parameter `eFuMo.clust(i).nu_fuzz`.

The aforementioned data set enables the reconstruction of the *efm* object. Obviously, the reconstructed object comprises less information than the original object, however, the selected data-set enables full derivation of the prediction model.

3. GRAPHICAL USER INTERFACE

The basic goal of the graphical user interface (GUI) for identification and conversion between *efm* object with eFuMo model for prediction and XML file (see Fig. 1) is to enable the user to work with eFuMo models easily and intuitively. GUI supports identification of eFuMo models from suitable data, validation of the identified models, and conversion of the models formulated as *efm* objects to XML files. The conversion from an *efm* object to an XML file and vice versa can be carried out via a DOM object (org.apache.xerces.dom), which allows for user manipulation of XML structure within Matlab (see Fig. 2).The conversion can be conducted directly as well.

The basic window of GUI is divided into four main panels (see Fig. 1): panel *Identification* for identifying an eFuMo model; panel *Validation* for validating an eFuMo model; panel *eFuMo* for manipulating an eFuMo model; and panel *DOM* for working with DOM objects and XML files.

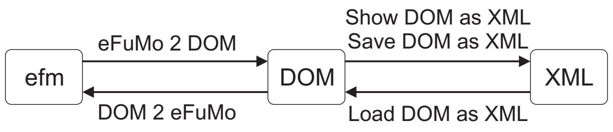


Fig. 2. Conversion options

Panel *Identification* includes a subpanel *Identification data* with two buttons. *Load a new structure* loads a new pre-formatted data structure, whereas *Import data* imports raw data from data files *.sim or *.cD that are used in production analysis. During the import operation the user sets a new sampling time, selects the relevant signals and sets their respective delays in the regressor. Finally, the data structure is saved as a data file that can be easily loaded in the GUI. The path to the selected active data structure is shown in the field on the top of the subpanel. The path can also be entered manually. From the next subpanel *Identification of eFuMo model* it is possible to run the eFuMo configuration tool. The user can select a segment of data for identification in the appropriate fields and start the identification algorithm with the *Start identification* button. The user can also plot the relevant signals using the *Plot data* button.

Panel *Validation* includes a subpanel *Validation data* with a button *Load a new structure* loads a new pre-formatted data structure for validation. The path to the selected active data structure is shown in the field on the top of the subpanel. The path can also be entered manually. In the next subpanel *Validation of eFuMo model* it is possible to select a segment of data for validation in the appropriate fields and start the validation algorithm with the *Start validation* button. The user can select between either a k-step prediction or a simulation. Here, k is a user defined parameter. After the validation is carried out, the results are plotted and the validation criteria MSE (mean squared error) (1) and NPE (normalized prediction error) (2) are calculated and displayed.

$$MSE = \frac{\sum_{k=1}^{N}(\hat{y}(k) - y(k))^2}{N} \tag{1}$$

$$NPE = \sqrt{\frac{\sum_{k=1}^{N}(\hat{y}(k) - y(k))^2}{\sum_{k=1}^{N}(y(k))^2}} \cdot 100\% \tag{2}$$

Panel *eFuMo* is made up of two subpanels: *Selected eFuMo* subpanel and *Conversion from eFuMo* subpanel. The *Selected eFuMo* subpanel allows the selection of the *efm* structure with the relevant eFuMo model. The selected active data structure is shown in the field on the top of the subpanel. It can be entered using the *Select a new eFuMo* button or manually. The *Conversion from eFuMo* subpanel includes two buttons. *eFuMo 2 DOM* button converts the selected eFuMo model into a DOM object, whereas *eFuMo 2 XML* button converts it directly into an XML file. The algorithm first gathers the relevant data of the fuzzy model for prediction and then organizes it in a DOM object or an XML file.

Panel *DOM* is made up of three subpanels: *Selected DOM* subpanel, *Conversion to eFuMo* subpanel, and *XML and DOM* subpanel. The *Selected DOM* subpanel allows the selection of the DOM structure with the relevant eFuMo model for prediction. The selected active object is shown in the field on the top of the subpanel. It can be entered using the *Select a new DOM* button or manually. The *Conversion to eFuMo* subpanel includes two buttons. *DOM 2 eFuMo* button converts the selected DOM object into an eFuMo model for prediction, whereas *XML 2 eFuMo* button extracts the *efm* object directly from an XML file. The *XML and DOM* subpanel includes three buttons. The buttons *Show DOM as XML* and *Save DOM as XML* enable obtaining an XML file from the selected DOM object and displaying or saving it, respectively. *Load XML to DOM* button loads an XML file and converts it into a DOM object.

4. EXAMPLE

Below we present an illustrative example of an XML formulation, which was obtained by identifying a fuzzy model of the Tennessee Eastman process simulation dynamics benchmark (see Downs and Vogel [1993]). Note that for clarity reasons the structure is is not fully expanded.

```
<?xml version="1.0" encoding="utf-8"?>
<PMML xmlns="http://www.dmg.org/PMML-4_1" xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-instance" version="4.1">
 <Header copyright="GorazdK">
  <Annotation>This is a prediction model derived from an eFuMo
   object. The XML formulation complies with the PMML standard
   (xmlns="http://www.dmg.org/PMML-4_1").</Annotation>
  <Timestamp>2013-07-08T03:10:23+02:00</Timestamp>
 </Header>
 <DataDictionary numberOfFields="6">
  <DataField dataType="double" displayName="F_p" name="F_p"
   optype="continuous"/>
  ...
 </DataDictionary>
 <Extension name="fuzzyModel">
  <FuzzyModel functionName="regression" modelName=
   "eFuMo prediction model" modelType="efuMoPredictionModel">
    <MiningSchema>
     <MiningField name="F_p"/>
     ...
     <MiningField name="OpCost"/>
    </MiningSchema>
    <RegressorDefinition numberOfInputs="11">
     <RegressorInput id="1" name="F_p(k-1)">
       <Delay value="1"/>
       <DerivedField dataType="double" optype="continuous">
        <FieldRef field="F_p"/>
       </DerivedField>
     </RegressorInput>
     <RegressorInput> ... </RegressorInput>
     ...
     <RegressorInput> ... </RegressorInput>
    </RegressorDefinition>
    <OutputDefinition numberOfOutputs="1">
     <FuzzyModelOutput id="1" name="OpCost(k)">
       <Delay value="0"/>
       <DerivedField dataType="double" optype="continuous">
        <FieldRef field="OpCost"/>
       </DerivedField>
     </FuzzyModelOutput>
    </OutputDefinition>
    <Rules numberOfRules="8">
     <Rule_i i="1">
       <IfPart>
        <MembershipFunctionDefinition type="cluster">
          <Cluster>
           <Array type="real"> ... </Array>
           <Covariances>
            <Matrix nbCols="12" nbRows="12">
             <Array type="real"> ... </Array>
```

```
              ...
              <Array type="real"> ... </Array>
            </Matrix>
          </Covariances>
        </Cluster>
      </MembershipFunctionDefinition>
     </IfPart>
     <ThenPart>
      <OutputFunctionDefinition type="localLinearModel">
        <LocalLinearModel>
         <Matrix nbCols="1" nbRows="12"> ... </Matrix>
        </LocalLinearModel>
      </OutputFunctionDefinition>
     </ThenPart>
   </Rule_i>
   <Rule_i i="2"> ... </Rule_i>
   ...
   <Rule_i i="8"> ... </Rule_i>
  </Rules>
  <GeneralParameters>
   <NumberOfOutputs>1</NumberOfOutputs>
   <MembershipDegreeCalculationParameters>
     <MembershipDegreeCalculatingMethod>
      Euclidean component distance with exponent function
     </MembershipDegreeCalculatingMethod>
     <FuzzinessFactor>2</FuzzinessFactor>
     <OverlappingFactor>1</OverlappingFactor>
   </MembershipDegreeCalculationParameters>
  </GeneralParameters>
 </FuzzyModel>
 </Extension>
</PMML>
```

5. CONCLUSION

The presented XML structure is compliant with the PMML standard and supports general fuzzy model formulations. The graphical user interface enables the user to manipulate with eFuMo models easily and intuitively. It provides user-friendly functions for quick and easy evolving identification of eFuMo models and simple tools for converting between *efm* objects, DOM objects and XML files.

The XML files that comply with the PMML standard provide a way for applications to describe and exchange models produced by data-mining and machine-learning algorithms. The goal of the presented work is to provide a platform for including fuzzy models into the PMML framework and thus stimulate interoperability of various applications that take advantage of fuzzy models, e.g. in production dynamics analysis and control, as well as in other implementations.

ACKNOWLEDGEMENTS

The work has been supported by the Competence Center for Advanced Control Technologies, which is partly financed by the Republic of Slovenia, Ministry of Higher Education, Science and Technology and European Union (EU) European Regional Development Fund.

REFERENCES

G. Acampora. Fuzzy markup language. 2013. URL `http://www.corisa.it/fml/Home.html`.

R. Babuška. *Fuzzy modeling and identification.* Technische Universiteit Delft, 1996.

R. Babuška. *Fuzzy Modelling for Control.* KAP, 1998.

R. Babuška and H. B. Verbruggen. An overview of fuzzy modelling for control. *Control Engineering Practice*, 4 (11):1593–1606, 1996.

M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg. How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective. *International jornal of mechanical, industrial science and engineering*, 8(1):37–44, 2014.

J. Castro. Fuzzy logic controllers are universal approximators. *IEEE Trans. System Man Cybernet*, 25:629–635, 1995.

CCACT. Competence centre for advanced control technologies. 2013. URL `http://www.kcstv.si/en/`.

DMG. Predictive model markup language. 2013. URL `http://www.dmg.org/v4 1/GeneralStructure.html`.

D. Dovžan and I. Škrjanc. Predictive functional control based on an adaptive fuzzy model of a hybrid semi-batch reactor. *Control Engineering Practice*, 18(8):979–989, 2010a.

D. Dovžan and I. Škrjanc. Recursive clustering based on a gustafsonkessel algorithm. *Evolving Systems*, 2(1):15–24, 2010b.

D. Dovžan and I. Škrjanc. Recursive fuzzy c-means clustering for recursive fuzzy identification of time-varying processes. *ISA Transactions*, 50(2):159–169, 2011.

J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. *Computers and Chemical Engineering*, 17(3):245–255, 1993.

D. Girimonte and R. Babuška. Structure for nonlinear models with mixed discrete and continuous inputs: a comparative study. In *Proc. of IEEE International Conf. on system, Man and Cybernetics*, pages 2392–2397, 2004.

F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

M. Glavan, D. Gradišar, M. Atanasijević-Kunc, S. Strmčnik, and G. Mušič. Input variable selection for model-based production control and optimisation. *The International Journal of Advanced Manufacturing Technology*, 68(9-12):2743–2759, 2013a. ISSN 0268-3768.

M. Glavan, D. Gradišar, S. Strmčnik, and G. Mušič. Production modelling for holistic production control. *Simulation Modelling Practice and Theory*, 30(0):1 – 20, 2013b. doi: http://dx.doi.org/10.1016/j.simpat.2012.07.010.

IEC. Enterprise-control system integration part 3: Activity models of manufacturing operations management, 2007.

IEC. Free fuzzy logic library. 2013. URL `http://ffll.sourceforge.net/fcl.htm`.

I. Škrjanc and D. Matko. Fuzzy predictive functional control in the state space domain. *Journal of Intelligent and Robotic Systems*, 31:283–297, 2001.

I. Škrjanc, S. Blažič, and O. Agamenonni. Identification of dynamical systems with a robust interval fuzzy model. *Automatica*, 41:327–332, 2005.

L. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

S. Zorzut, V. Jovan, D. Gradišar, and G. Mušič. Closed-loop control of a polymerisation plant using production performance indicators (pi). *Int. Journal of Computer Integrated Manufacturing*, 22(12):1128–1143, 2009.