Software interfaces for interaction of intelligent fuzzy and neuro-fuzzy models in an end-point software risk-management system

A V Senkov[1] **and Yu A Senkova**[1]

[1] Computers department, National Research University Moscow Power Engineering Institute, Smolensk branch, Smolensk, Russian Federation

Abstract. In this work a software interface for interaction of intelligent fuzzy and neuro-fuzzy models is proposed. It includes an intelligent models design pattern, consisting of an intelligent model pattern, a pattern for implementing hybrid models and a pattern for working with numbers of different nature; a pattern of the database structure that supports the intelligent model design pattern. An example of a system implemented using the designated patterns is given. The implemented system has provided the variability of the use of different models, and provided the opportunity to change the set and sequence of model calls without involving a programmer. The proposed software interface, firstly, is universal and provides software implementation of various types of intelligent models. Secondly, it provides all the main types of hybridization of intelligent models. Thirdly, the interface allows to build highly adaptable intelligent systems, including systems that are built by a user without involving a programmer. Fourthly, this interface provides ample opportunities for the exchange of best practices, which is understood as the ability not only to transmit facts of the knowledge base, but also knowledge base models or rules.

1. Introduction

Now, a significant number of intelligent models and systems have been developed that solve both individual problems and complex problems characterized by inaccuracy and uncertainty of system and external parameters, for example [1, 2, 3]. As a rule, such works are aimed at solving particular cases of problems. However, the world around us is characterized by a high degree of variability, which results in models and software that recently performed the assigned tasks effectively becoming obsolete and not suitable for solving new interpretations of such problems, or solving problems in new conditions. Thus, it is obvious that modern intelligent systems should provide the possibility of flexible adaptation, restructuring under changing conditions.

One of the modern approaches to creating flexible efficient intelligent systems with a high degree of adaptability is the approach based on hybridization of neural network and fuzzy models. The following main types of hybridization of such models were considered in [4-6]:

- hybridization with functional substitution - one technology (model) is taken as the dominant one, and its individual components are replaced by components of other technologies (models);
- hybridization with interaction - technologies (models) are used relatively independently, and while exchanging information, they perform various tasks to achieve a common goal;

- polymorphic hybridization - one technology (model) is used to simulate and implement the function of another.

Thus, modern intelligent systems, in addition to the requirement of effective solution of the assigned task, must meet the following requirements:

- having a high degree of adaptability to changing external conditions of the model application and changing structure and functions of the object within which they are used;
- providing ample opportunities for hybridization of neural network and fuzzy models.

The task of such a scale can be solved only by developing common "rules of the game" in the software implementation of intelligent models, namely, development of a designing pattern [7-9] of intelligent models, including the intelligent model software interface and the typical database structure that ensures the functioning of such a model.

2. Intelligent model design pattern

2.1. Interface for implementing intelligent models
Generally, an intelligent model can be represented as a tuple:

$$R = < I, O >,$$

where I is a vector of the intelligent model inputs;
O is a vector of the intelligent model outputs.
The input and output vectors of intelligent models define the list of public properties of the intelligent model interface. Figure 1 shows an intelligent model interface.

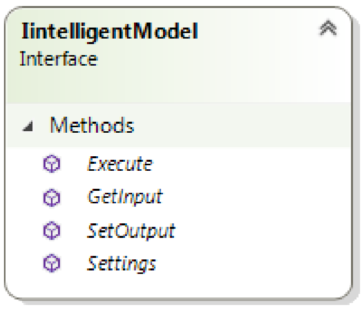

Figure 1. Intelligent model interface.

Thus, all models based on the interface IIntelligentModel will have the following common methods.

- GetInput – a method that receives input parameters for the model. As a result of the method execution, the requesting system receives a set of parameters that can be used as inputs of the model. This set includes all the parameters that you need to define for calculating the model. These parameters include the input parameters of the model itself, as well as all the input parameters of the related models, the results of which are used as inputs for the model in question.
- SetOutput – a method that defines the output parameters of the model. As a result of this method, a set of parameters that must be calculated and used as model outputs is specified for the model. This list of output parameters can be further used as inputs for other models.
- Execute – a method that allows calculating the output parameters of a model using the defined input parameters.

- Setting – a method that allows for learning or setting up models. This method, as a rule, should call a separate user interface for setting up the model. Typically, the configuration is carried out in 3 stages. Firstly, the structure of the model is constructed. To do this, a source file, presented in a standard format (for example, in the format of graphical risk notation proposed in [10] or notation of fuzzy business processes [11]) must be forwarded to the input of the method. The source data is converted into the model structure as a result of applying the learning method or constructing a programmatically implemented model. Secondly, optionally, the graphic adaptation of the model is carried out (if assumed by the model itself). The model image is brought by the user into a human-readable view. Thirdly, if necessary, operational simulation is performed. For this, the user forwards test data to the model input and compares the actual result with the expected result. If necessary, the model can be further developed (trained, or re-trained).

2.2. Class for implementing hybrid models
The hybrid model should also be built on the basis of the developed interface to ensure compatibility with other models.

The hybrid model class may be specified the following way (figure 2).
The class HybridModel has the following properties:

- ListInput – input parameter list;
- ListOutput – output parameter list;
- ListMathModel – list of mathematical models used in the hybrid model in the order of their launch.

In addition to the interface methods already discussed, the class defines the method AddMathModel, which allows you to add another model to the hybrid model.

Based on the presented class, models using the following types of hybridization can be defined: with functional substitution, with interaction, and polymorphic hybridization.

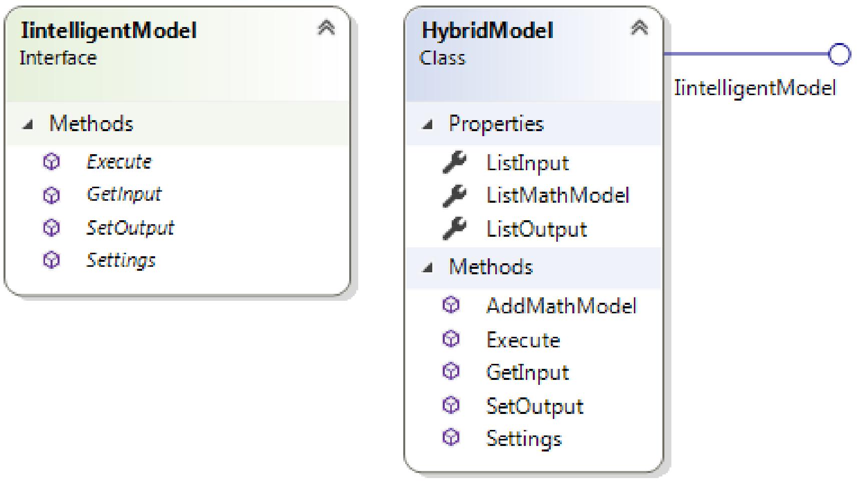

Figure 2. Definition of hybrid model class.

2.3. Pattern for presenting clear and fuzzy numbers
Another important element in defining the interface for hybridization of intelligent models is the pattern of presentation of clear and fuzzy numbers. Different models can often operate simultaneously with both clear and fuzzy numbers. In order to unify the data exchange, it is necessary to develop an appropriate design pattern.

For these purposes, the universal class NumberContrainer has been developed, which has the following properties:

- ValueTypeId - defines the number type (clear / fuzzy);
- ValueString - the string in which the serialized object - a descendant of the abstract class BaseNumberValue - is stored.
- Number - deserialized object - an instance of a descendant of the abstract class BaseNumberValue.

The class NumberClear describes a clear number and contains only the value of the clear number Value.

The class NumberFuzzy describes a fuzzy number with a membership function of trapezoidal or triangular type and contains the following fields describing it:

- A - left value of the carrier;
- B - right value of the carrier;
- Alpha – the value of the left border of a fuzzy number;
- Beta – the value of the right border of a fuzzy number.

In a similar way, classes of fuzzy numbers can be developed that are defined by membership functions of an arbitrary kind, such as parametric (Gaussian, Z-shaped, S-shaped, U-shaped), as well as piecewise-analytic membership functions and membership functions that are defined pointwise (by sets of points).

Use of such a pattern ensures the model independence while working with both clear and fuzzy numbers. A diagram of the pattern classes is shown in figure 3.

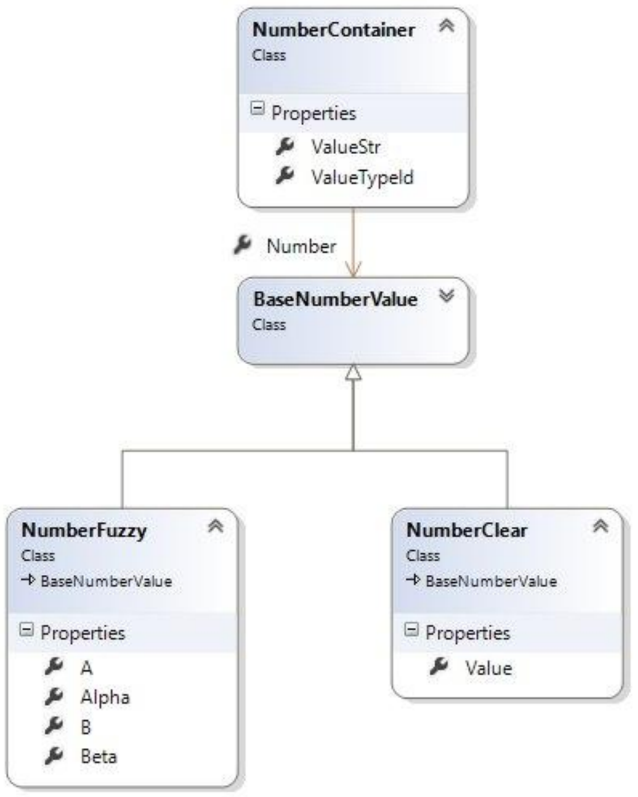

Figure 3. Pattern class diagram for presenting clear and fuzzy numbers.

3. Pattern of the database structure for implementation of intelligent models

The proposed pattern of intelligent model design should be supported by the pattern of the database structure. A typical database structure for implementation of intelligent models is shown in figure 4.

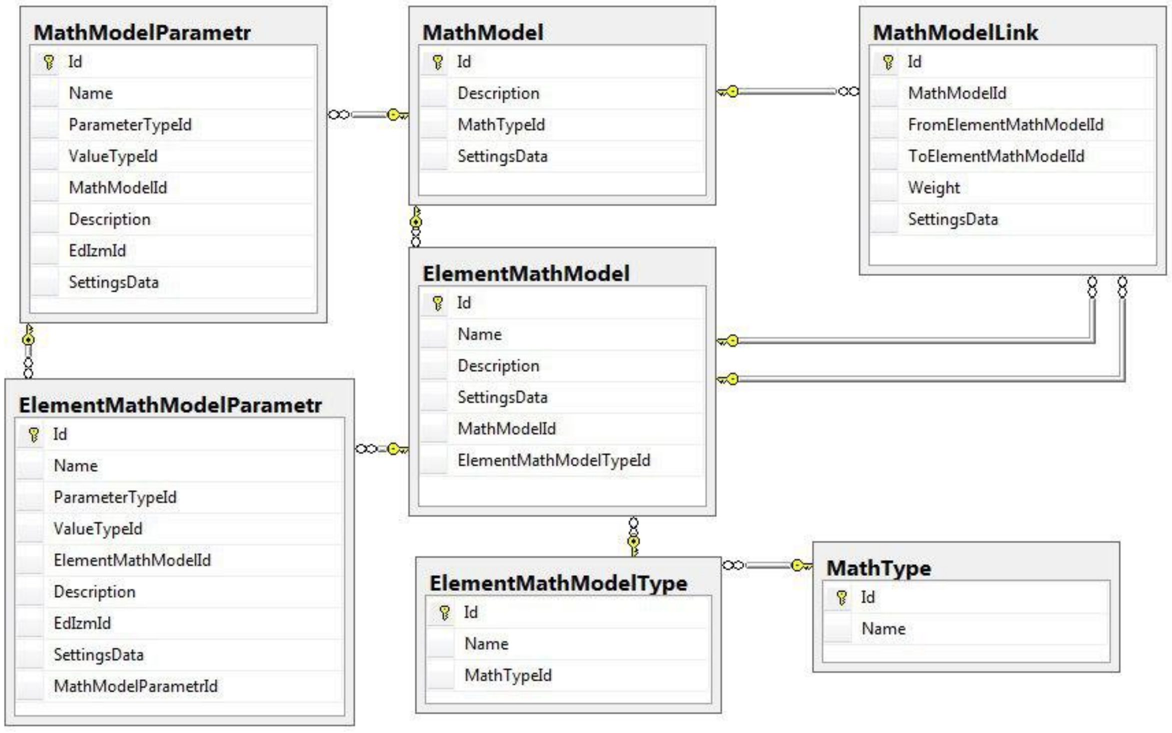

Figure 4. Database structure pattern for implementation of intelligent models.

In the presented structure, the table MathModel stores information about all models as a whole, as well as their type (for example, fuzzy fault tree). This table also stores information about which models are used in the hybrid model, and the sequence of their execution. The table ElementMathModel stores information about all elements of mathematical models (elements, concepts, rules, terms, etc.). The table ElementMathModelType stores the decoding of the model elements names, and the table MathType stores the decoding of the models names. The table MathModelPatametr stores information about all parameters of the mathematical model (parameter type, unit, value type). The table ElementMathModelParametr stores information about the parameters of specific model elements, and the table ElementValueMathModelParametr stores information about parameters for specific values (terms) of mathematical models elements. The table MathModelLink stores information about all links between model elements, namely from which element the link comes, to which element the link arrives, as well as the link weight, if any.

4. Approach to hybridization of models

The above software interface for hybridization of intelligent fuzzy and neural-fuzzy models ensures implementation of all the previously listed hybridization types.

Hybridization with functional substitution is implemented by building new models using old ones in its structure. An example of such hybridization is shown in figure 5.

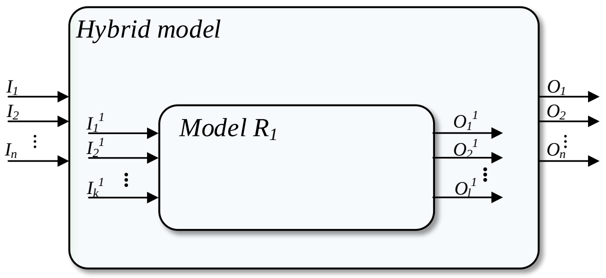

Figure 5. Example of a hybrid model with functional substitution.

Hybridization with interaction is implemented by forming a complex model linking inputs of some models with outputs of other. The complex model itself is built in the image and likeness of base models, which allows to build multi-level hierarchical structures of models (figure 6).

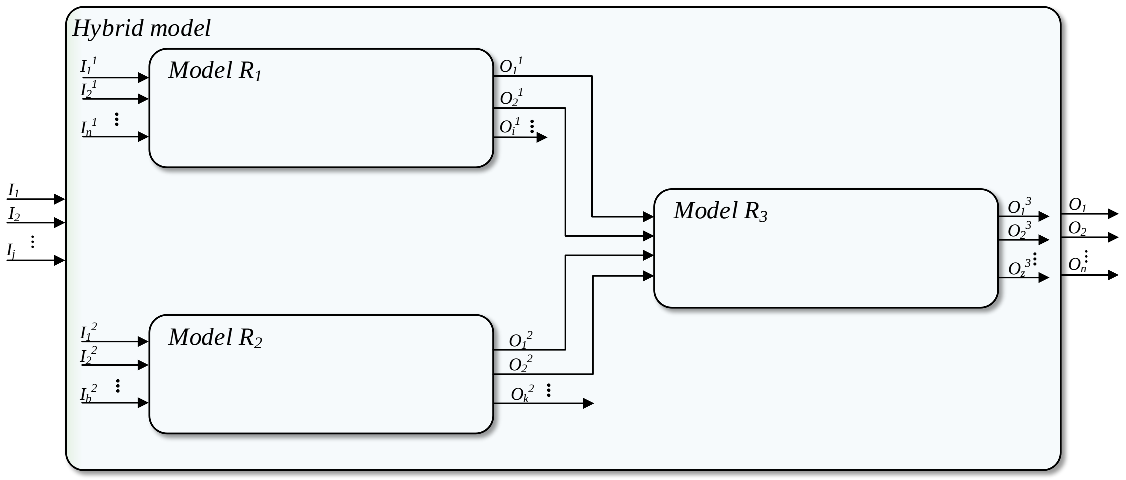

Figure 6. Example of a hybrid model with interaction.

Polymorphic hybridization, as well as hybridization with functional substitution, is implemented through building new models by changing the functioning algorithms of older models.

5. Approbation of the software interface for hybridization of intelligent fuzzy and neural-fuzzy models

The presented software interface was tested in the implementation of a decision support system for preliminary management of operational risks of a computational cluster. Within the framework of this project, the following models were successfully implemented:

- fuzzy fault tree;
- fuzzy Bayesian network;
- fuzzy cognitive map;
- fuzzy production model;
- neuro-fuzzy classifier.

The hybrid model was built through hybridization with functional substitution and interaction. The general scheme of the hybrid model is shown in figure 7.

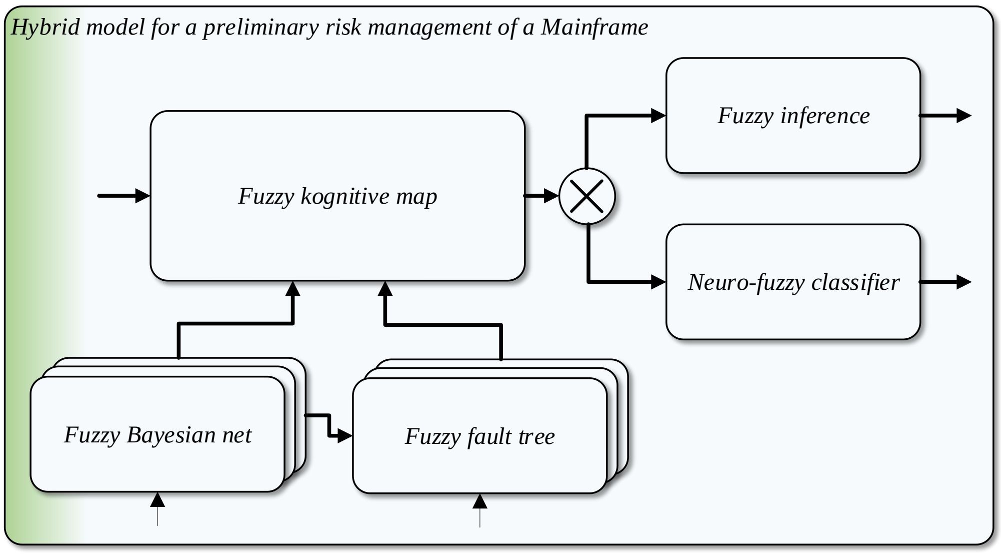

Figure 7. Hybrid neuro-fuzzy model, based on proposed software interface.

The developed system provides decision support for preliminary risk management with regard to the three main aspects of risks [12]: process, structural and systemic. In addition, the system implements all the main stages of risk management: identification, analysis, evaluation and development of management decisions.

The developed system provides the user with the possibility of using various models. So, at the final stage of management, either a fuzzy production model or a neural-fuzzy classifier could be used. In addition, the system allows, without involving a programmer, to change the set and sequence of model calls. As unchangeable or basic (due to a number of functional limitations), a fuzzy cognitive map model was chosen.

6. Conclusion

Thus, the proposed software interface, firstly, is universal and provides software implementation of various types of intelligent models having extremely different methods of application (both static models and dynamics models). Secondly, it supports all the main types of hybridization of intelligent models. Thirdly, the interface allows you to build intelligent systems with a high degree of adaptability, including systems that are built by a user without involving a programmer. Fourthly, such an interface provides ample opportunities for the exchange of best practices, which is understood as the possibility not only to transmit facts of the knowledge base in terms of D. Pospelov between intelligent systems, but also to transfer models (transfer of knowledge base rules).

7. References

[1] Denisenkov M A 2013 Application of adaptive fuzzy situation networks for solving analytical problems of decision support *Computer mathematics systems and their applications* **14** pp 72-75

[2] Kozlov P Yu 2017 Methods of automated analysis of short unstructured text documents // *Software products and systems* **1** pp 100-105

[3] Zakharov A S 2014 Temporal conclusion using fuzzy Bayesian networks *Smolensk State University Bulletin* **1 (25)** pp 417-439

[4] Averkin A N and Prokopchina S V 1997 Soft calculations and measurements *Intelligent systems* **2, 1-4** pp 93-114

[5] Borisov V V 2014 Hybridization of Intellectual Technologies for Analytical Tasks of Decision-Making Support *Journal of Computer Engineering and Informatics* **Vol. 2 Iss. 1** pp 148-156

[6] Komartsova L G 2003 *Research of neural network and hybrid methods and technologies in intellectual systems of support of decision-making* (Kaluga branch of Moscow State Technical University. N.E. Bauman)
[7] Grady Booch 2007 *Object-oriented analysis and design with examples of applications* (Williams)
[8] Gamma E, Helm R, Johnson R and Vlissides D *Methods of object-oriented design. Design Patterns* (Peter)
[9] Teplyakov S 2016 *Patterns of designing on the .NET platform* (Peter)
[10] Senkov A V 2016 Graphical notation for the presentation of the process of managing complex risks *Modern high technology* № **12-1** pp 72-81
[11] Sorokin E V, Senkov A V, Margolin M S 2016 Formal language for describing fuzzy business processes *Scientific almanac* № **10-3 (24)** pp 278-284
[12] Senkov A V 2016 *Risk management: intellignet models, methods, tools* (Universum)

Acknowledgments
The reported study was funded by RFBR, according to the research project No. 16-37-60059 mol_a_dk.