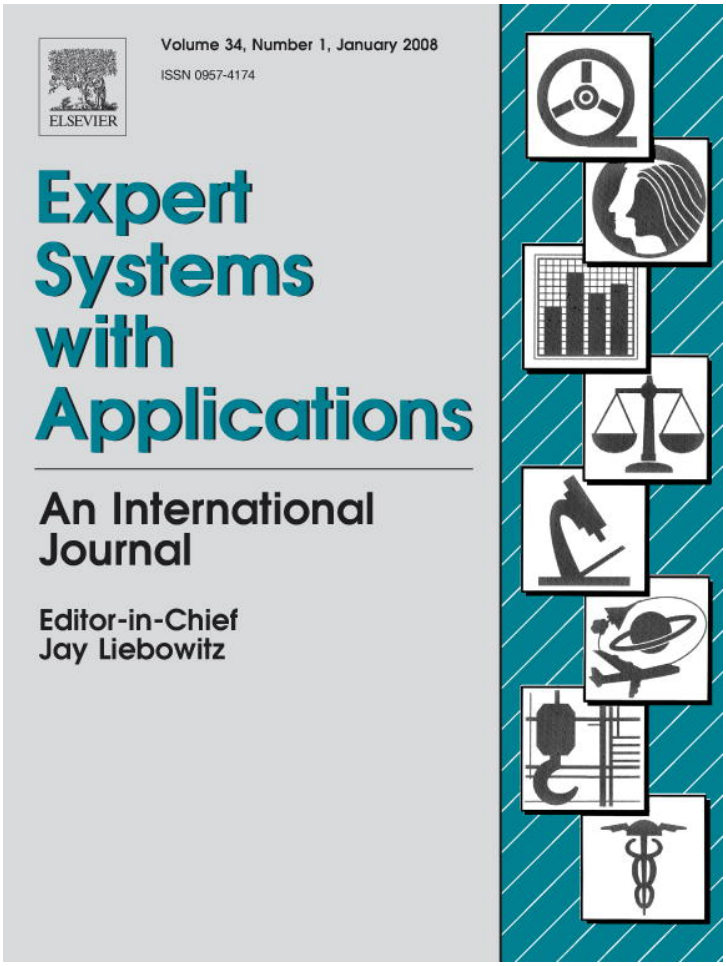

Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 34 (2008) 796–804
Expert Systems with Applications
www.elsevier.com/locate/eswa

Java Fuzzy Kit (JFK): A shell to build fuzzy inference systems according to the generalized principle of extension

Omar López-Ortega

Universidad Autónoma del Estado de Hidalgo, Instituto de Ciencias Básicas e Ingeniería, Centro de Investigación en Tecnologías de Información y Sistemas, Carretera Pachuca - Tulancingo Km. 4.5, Pachuca, Hidalgo, México

Abstract

In this article the author presents JFK, which stands for Java Fuzzy Kit. JFK is an Application Programming Interface (API) that complies with both, a general structure of a fuzzy rule base and the necessary processing to compute the generalized principle of extension. A recurrent structure is found for a class of fuzzy expert systems, known as the Mamdani model. This leads to claim that a design pattern exists, since core objects, which are present regardless the specific application are identified. However, there is not a general shell to build fuzzy expert systems, and this provokes that current fuzzy expert systems are build on an ad-hoc basis. The modelling of JFK is done according to the Unified Modelling Language specifications. Along with the UML modelling three important algorithms are described, which serve to perform the generalized principle of extension. The usage of JFK is illustrated with an example, namely the ranking of swimmers. Preliminary results on this study case are the basis to propose the realization of fuzzy distributed decision systems. This goal is accomplished by providing agents with fuzzy expert systems, with the integration of JFK and the standardized platform JADE (Java Agent Development Environment).

Keywords: Fuzzy expert systems; Object-oriented modelling; Mamdani inference; Java fuzzy kit

1. Introduction

Fuzzy logic represents a cornerstone in the development of computer systems intending to mimic an aspect of human inference: it has been claimed that the theory introduced by Zadeh (Zadeh, 1973) helps computing vagueness. Nowadays fuzzy logic is both a philosophy and a set of well-structured algorithms to model and compute complex systems. From a software design point of view, Artificial Intelligence in general, and Fuzzy Logic in particular, represents a challenge to build intelligent applications such as expert systems. However, current literature mostly reports on the design of boolean expert systems. Ling, Tseng, and Tsai (2003) present an object-oriented model which has the capability of learning and thinking. Souza and Ferreira (2002) offer a survey on object-oriented design of rule-based systems. This approach, though valuable in terms of software engineering, has not been widely followed for designing fuzzy expert systems. Shu-Hsien (2005) reviews the methodologies that have been employed to design and construct expert systems, and concludes that sound fuzzy expert systems can be implemented on the object-oriented approach.

In this article the author presents a Java Fuzzy Kit (JFK). JFK is an application programming interface (API) that complies with both, a general structure of a fuzzy rule base and the necessary processing to compute the generalized principle of extension. Numerous benefits can be encountered. It can be used on any platform, because it is implemented in the Java language and more importantly, it is the basis to actually develop fuzzy distributed decision making systems, under the multi-agent system paradigm. This line of research is quite promising as it will be shown in the paper. Despite the importance and power of fuzzy inference systems, to the best of the author's

E-mail address: lopezo@uaeh.reduaeh.mx

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.10.019

knowledge, this is the first attempt to model and implement an open and general fuzzy system, conforming to the generalized principle of extension.

The organization of the paper is as follows. Theoretical foundations as well as implementations of fuzzy expert systems are described in Section 2. Next, the performed analysis of requirements is contained in Section 3. The formal object-oriented models are outlined in Section 4. To continue, the validation of the system is shown in Section 5. Promising future work is outlined in Section 6, followed by concluding remarks.

2. Theory and implementation of fuzzy inference systems

Since there are numerous references on fuzzy logic basics, attention is placed on a particular structure of fuzzy expert systems known as the Mamdani model. In such a structure both, antecedent and consequent clauses are fuzzy propositions (Perfilieva, 2006; Yen & Langari, 1999). The Mamdani model extends the classic modus ponens rule, which is formally stated as follows (Stoll, 1979):

$$\prec A \wedge (A \rightarrow B) \rightarrow B \tag{1}$$

Eq. (1) means that given a rule that holds true ($A \rightarrow B$), and input A, then B holds true as well. The fuzzy logic generalized principle of extension is:

$$\prec A' \wedge (A \rightarrow B) \rightarrow B' \tag{2}$$

In this case, the input might be A or something similar to A (A'), yielding B or something quite similar to $B(B')$.

Thus, the rule ($A \rightarrow B$) takes the form:

IF x is A **THEN** y is B (3)

Such a rule generalizes the dependency relationships between variables. With the previous representation, any input that matches the linguistic variable represented by fuzzy set A, yields an output represented by membership function B. In its generalized form, the inputs are represented by crisp values, which live in a realm known as the Universe of Discourse (UoD), such that UoD $\subset \mathbb{R}$. It is well-known that in the vast majority of engineering applications, both inputs and outputs are numerical crisp values. The power of fuzzy expert systems lies in their capability for processing multiple input variables. Fuzzy expert systems are those which variables are related by means of fuzzy IF–THEN rules. Thus, the Mandani model of fuzzy IF–THEN rules is a fuzzy rule-based system, which complies with the following structure:

$$\begin{aligned}
&r_1: \textbf{IF } x_{11} \text{ is } A_{11} \textbf{ AND } x_{12} \text{ is } A_{12} \textbf{ AND} \ldots \textbf{AND } x_{1n} \text{ is } A_{1n} \textbf{ THEN } y \text{ is } B_1\\
&r_2: \textbf{IF } x_{21} \text{ is } A_{21} \textbf{ AND } x_{22} \text{ is } A_{22} \textbf{ AND} \ldots \textbf{AND } x_{2n} \text{ is } A_{2n} \textbf{ THEN } y \text{ is } B_2\\
&\ldots\\
&r_m: \textbf{IF } x_{m1} \text{ is } A_{m1} \textbf{ AND } x_{m2} \text{ is } A_{m2} \ldots \textbf{AND } x_{\mathrm{mn}} \text{ is } A_{\mathrm{mn}} \textbf{ THEN } y \text{ is } B_m
\end{aligned} \tag{4}$$

where:

r_i is the j_{th} rule ($j = 1, 2, \ldots, m$)
A_{ik} is the k_{th} antecedent clause of rule j ($k = 1, 2, \ldots, n$)
B_j is the consequent clause of rule j
x_{ik} is the k_{th} crisp input of rule j
y_i is the correspondent crisp output of rule j

According to Babuska (1998), fuzzy sets A_{ik} define fuzzy regions in the input space, for which the respective consequent is true. In the Mamdani model, the rules form a mapping between input and output variables. This is accomplished by an inference mechanism that, according to specific knowledge depicted in the fuzzy rule base and given input values, it derives a corresponding output value. The principle of extension is applied as the inference mechanism.

The inference algorithm based on the principle of extension, also known as the Mamdani inference, is (Babuska, 1998):

1. For each rule j, compute the degree of fulfillment β_j. This is done by using the minimum t-norm.
$$\beta_j = \mu_{A_{j1}}(x_1) \wedge \mu_{A_{j2}}(x_2) \wedge \ldots \wedge \mu_{Ajn}(x_n) \tag{5}$$
2. For each rule, derive the output fuzzy set B'_i, using the minimum t-norm.
$$\mu_{B'_j}(y) = \beta_j \wedge \mu_{B_j}(y), \forall y \in Y \tag{6}$$
3. Compute the aggregated output fuzzy set, by taking the maximum s-norm of the output fuzzy sets B'_i.
$$\mu_{B'}(y) = \bigcup_{j=1}^{m} \mu_{B'}(y), \forall y \in Y \tag{7}$$

Both, the structure and the algorithm of the Mamdani model are well-known for practitioners and researchers. However, fuzzy inference systems have been constructed according to specific domain problems. Juang, Lin, and Kao (2007) reports CRISP, which is a framework for a customer requirement information system, integrating a fuzzy rule-based system.

Bigus and Bigus (2001) report a fuzzy expert system of the additive type, fully developed in the Java language. The component approach has been explored to design fuzzy systems (Sendelj & Devedzic, 2004). Also, Horng, Lee, and Kuo (2003) describe an object-oriented framework for designing fuzzy knowledge systems; while an early approach by Wong and Chun (1999) provide object models of fuzzy sets.

FuzzyJ Toolkit (Orchard, 2006) is an API that can be used standalone to create fuzzy rules. It can also be used with JESS, the Expert System Shell from the Sandia National Laboratories (Friedman-Hill, 2003). Thus, it is currently known as Fuzzy JESS, since it is the fuzzy logic extension of JESS. Rules on FuzzyJ Toolkit adhere to the Tsukamoto structure. In the Tsukamoto fuzzy model the consequent of each rule is a fuzzy set with monotonical membeship function. However, this structure is not widely used because it is not as transparent as the Mamdani model

(Jan, Sun, & Mizutani, 1997, p. 84). This is a clear advantage of JFK over FuzzyJ Toolkit. The design process of JFK is shown next.

3. Analysis of requirements

As the purpose of this research is to construct an open object-oriented fuzzy kit, it is pivotal to clearly specify what it must do. Hence, the specifications for the desired Java Fuzzy Kit are the following:

1. JFK must be suitable for creating different fuzzy sets according to the matematical formulae for membership functions, such as Triangular, Trapezoidal, Sigmoidal and Gaussian.
2. JFK must be capable to define antecedent clauses, each of which can be represented by a different fuzzy set.
3. JFK must be helpful to define consequent clauses, each of which can be represented by a different fuzzy set.
4. Fuzzy rules in JFK must be formed by one or more antecedent clauses, and by only one consequent clause.
5. JFK must provide the capability to define a fuzzy rule base, consisting of one of more fuzzy rules.
6. JFK must accurately perform an inference process according to the principle of extension, independently of the fuzzy rule base to be processed.
7. JFK must provide the capability to acquire crisp inputs for the rules.

3.1. The underlying structure of JFK

By adhering to this set of principles, we are provided a set of distinguishable objects. As it can be extracted from the previous specifications, from which a list of core objects is presented in Table 1. A recurrent design structure is found for a fuzzy expert system conforming to the principle of extension. This leads to claim that a design patter exists, since we identify objects that are repeated regardless the specific application (Gamma et al., 1995). The proposed design is represented in Fig. 1.

Table 1
Objects within JFK

Object	Preliminary usage
Fuzzy set	To define fuzzy sets according to membership functions
Triangular	To define a triangular fuzzy set
Trapezoidal	To define a trapezoidal fuzzy set
Gaussian	To define fuzzy set based on the gaussian formula
Sigmoidal	To define a fuzzy set according to the sigmoidal function
Mamdani Clause	To create either an antecedent or a consequent clause, according to the fuzzy set needed
Mamdani rule	To set up a rule according to the Mamdani structure, with one or more antecedents, and one consequent
Mamdani Rule Base	To construct a fuzzy rule base, integrating one of more Mamdani rules
Min-max	To calculate either the t-norm or the s-norm, employed during the inference process

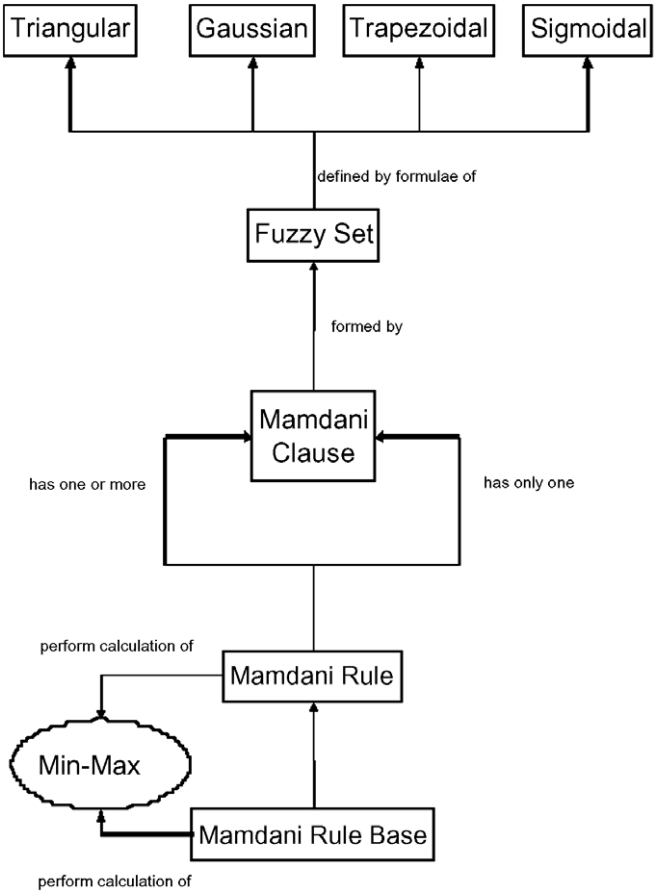

Fig. 1. Design patter of Java Fuzzy Kit.

3.2. The dynamics within

The next step in the requirement analysis phase consists of translating the Mamdani algorithm to programming units. This is necessary to set up the methods that must be coded so that objects of Table 1 coordinate, and derive the right output value. So, Eqs. (5)–(7) are viewed as the three algorithms described next.

Algorithm 1. Degree of fulfillment

```
INPUT: two different real numbers mem1,
mem2, result of the computation of member-
ship functions
OUTPUT: a real number beta that represents
the minimum of the membership functions

1. set beta ← 0
2. Determine the number n of antecedent
   clauses for each rule
3. for i = 1,2 ... n do the following
     beta=min(mem1, mem2)
     return (beta)
```

Algorithm 2. Output fuzzy set B'_i

```
INPUT: a real number beta that corresponds to
the degree of fulfillment, and a vector B of
real numbers that represents the fuzzy set
of the consequent
OUTPUT: a vector B' of real numbers
1. Determine the size of B
2. For the rule base, determine the number of
   rules m
3. for k from 1 to m
4. for i from 1 to size of B do the following
     B' [i] ← min(beta, B[i])
5. return (B')
```

Algorithm 3. Aggregated fuzzy set

```
INPUT: two vectors of real numbers B_j,B_{j+1},
that represent the output fuzzy sets of
rules j, j + 1; an integer size that represents
the cardinality of the universe of dis-
course of the consequent; and an integer rules
that represents the number of rules in the
rule base
OUTPUT: a vector of real numbers Aggregated,
that represents the aggregated output set
1. for i from 1 to rules
2. for j from 1 to size
     Aggregated=max(B_j, B_{j+1})
3. return (Aggregated)
```

Once the set of objects and the algorithms have been established, it is necessary to define the corresponding object-oriented model.

4. Software modelling

The modelling phase is done according to the Unified Modelling Language specifics (Fowler, 2004). Two formal models are presented. The first is the Class Diagram, and the other is the Sequence Diagram. The Class Diagram reflects the structure of the fuzzy expert system, whereas the Sequence Diagram mirrors the sequence of events that take place in order to complete the Mamdani inference. The Class Diagram is seen in Fig. 2. Fig. 3 is the Sequence Diagram.

4.1. Class diagram

The developed object-oriented model of Fig. 2 fully complies with the general pattern previously seen. First, a class `Base_Reglas_Mamdani` (Mamdani_Rule_Base) has a one-to-many association with class `regla_mamdani` (Mamdani_Rule). This allows a given rule base to be formed by one or many rules. Class `regla_mamdani`, in turn, is double related to class `Clausula_Mamdani`

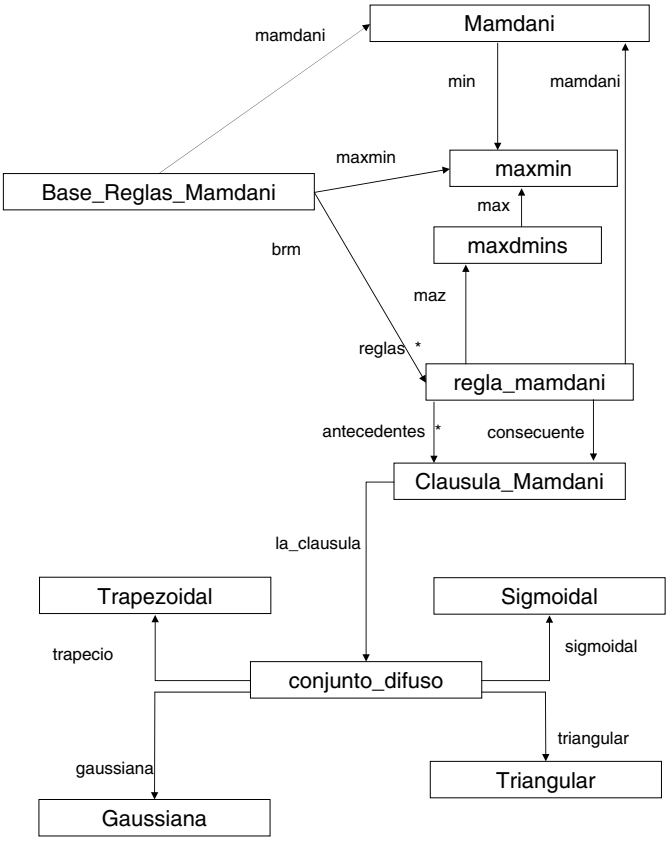

Fig. 2. Class Diagram.

(Mamdani_Clause). In the first association, one mamdani rule is related to one or many Mamdani Clauses. This is done to build the antecedents of the rule. The consequent of any rule is formed by instantiating only one Mamdani Clause. However, the particular membership function for either the antedent clause or the consequent clause is obtained via the class `conjunto_difuso` (fuzzy set), which is related to four types of membership functions (Trapezoidal, Sigmoidal, Gaussian and Triangular). Classes `Trapezoidal`, `Sigmoidal`, Gaussiana and `Triangular` implement methods to construct the corresponding fuzzy sets. Classes `Mamdani`, `maxmin`, `maxdmin` are auxiliary objects to help compute the Mamdani inference. To fully understand this computational process, the sequence diagram is presented next.

4.2. Sequence diagram

The Sequence Diagram is divided in three stages. The actor is any application that incorporates fuzzy reasoning. Firstly, it is necessary to instantiate the following objects: `conjunto_difuso` (fuzzy set), `Clausula_Mamdani` (Mamdani clause), `regla_mamdani` (Mamdani rule) and `Base_Reglas_Mamdani` (Mamdani rule base). This is done to define the structural objects of the fuzzy rule base. The computational part starts soon after the Mamdani rule base is instantiated. A variable called

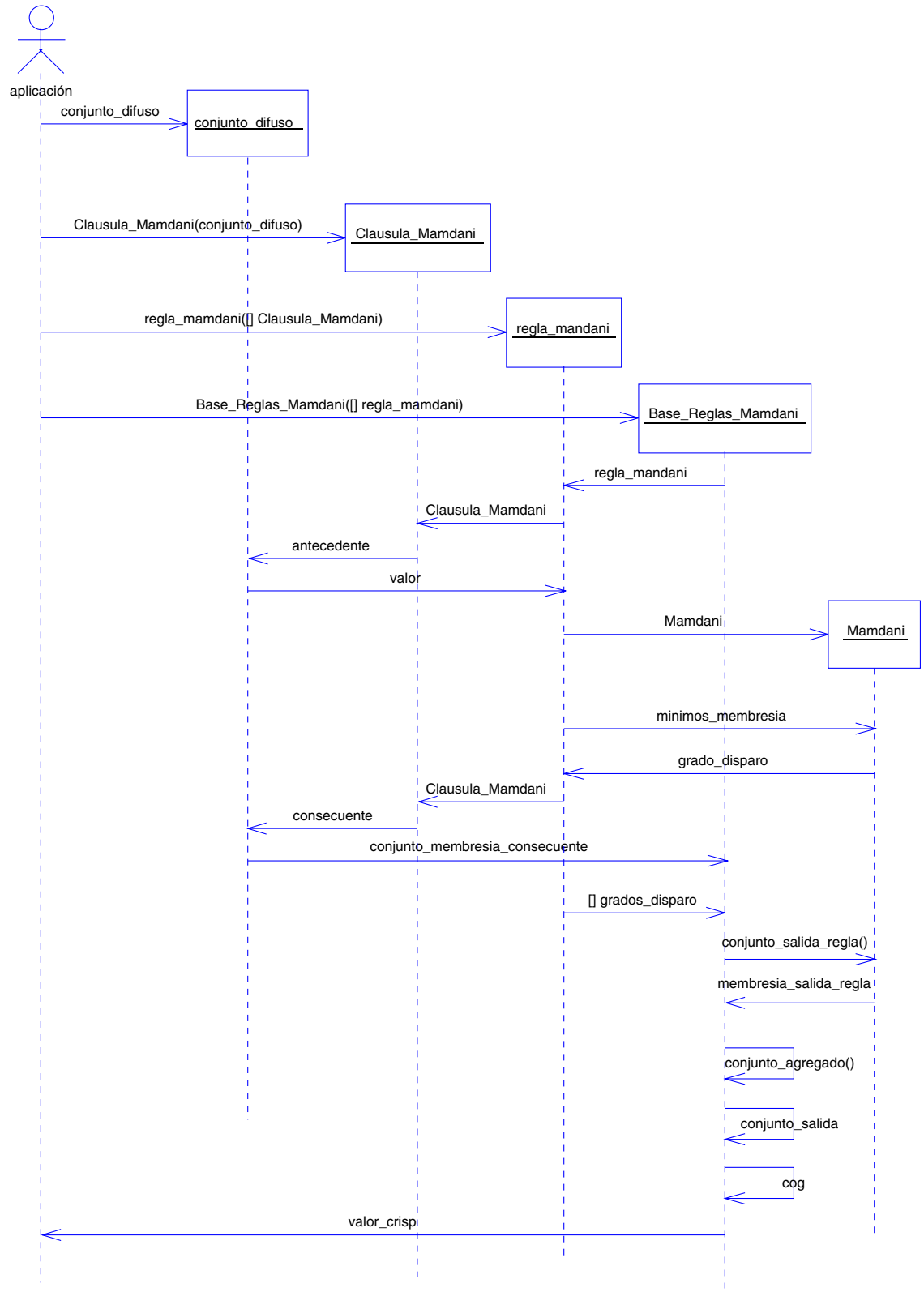

Fig. 3. Sequence Diagram.

`valor` contains the result of the t-norm calculation, yielding the minimum value of the antecedents, for each rule. Next, the `Mamdani` class is instantiated. This class receives two values, one double and an array of double values. The method `conjunto_salida(double, double[])` determines the output set for each rule. It is necessary to calculate the aggregated fuzzy set for the entire rule base. This is accomplished through a series of method calls

```
public class conjunto_difuso
{       double[] conjunto_generado = new double [1101];
        double valor_membresia; int i;
                Triangular triangular = new Triangular();
                Trapezoidal trapecio = new Trapezoidal();
                Gaussiana gaussiana = new Gaussiana();
                Sigmoidal sigmoidal;
                Bell bell = new Bell();
                String forma = new String();

conjunto_difuso(String forma){this.forma = forma;}

double[] triangular(int uodi, int uodf, double a, double b, double c){
 for(i = uodi; i<= uodf; i++){
  conjunto_generado[i] = triangular.triangular_membresia(i, a, b, c);
 }
 return conjunto_generado;
}
double triangular(double punto, double a, double b, double c){
 valor_membresia=triangular.triangular_membresia(punto, a, b, c);
 return valor_membresia;
}
double[] trapezoidal(int uodi, int uodf, double a, double b, double c, double d){
 for(i = uodi; i<= uodf; i++)
 {
  conjunto_generado[i] = trapecio.trapezoidal_membresia(i, a, b, c, d);
 }
 return conjunto_generado;
}
 double trapezoidal(double punto, double a, double b, double c, double d){
 valor_membresia = trapecio.trapezoidal_membresia(punto, a, b, c, d);
 return valor_membresia;
}
double[] gaussiano(int uodi, int uodf, double media, double desv){
 for(i = uodi; i<= uodf; i++)
 {
  conjunto_generado[i] = gaussiana.gaussiana_membresia(i, media, desv);
 }
 return conjunto_generado;
}
double gaussiano(double punto, double media, double desv){
 valor_membresia = gaussiana.gaussiana_membresia(punto, media, desv);
 return valor_membresia;
}
double[] sigmoidal(int uodi, int uodf, double a, double c){
 for(i = uodi; i<= uodf; i++)
 {
  conjunto_generado[i] = sigmoidal.sigmoidal_membresia(i, a, c);
 }
 return conjunto_generado;
}
double sigmoidal(double punto, double a, double c){
 valor_membresia = sigmoidal.sigmoidal_membresia(punto, a, c);
 return valor_membresia;
}
double[] bell(int uodi, int uodf, double a, double b, double c){
 for(i = uodi; i <= uodf; i++)
 {
  conjunto_generado[i] = bell.bell_membresia(i, a, b, c);
 }
 return conjunto_generado;
}
double bell(double x, double a, double b, double c){
 valor_membresia = bell.bell_membresia(x, a, b, c);
 return valor_membresia;}}
```

Fig. 4. Generation of fuzzy sets.

between classes `Base_Reglas_Mamdani` and `Mamdani`. The array of double values `conjunto_salida` is the aggregated fuzzy set. The crisp value is obtained via the calculation of its center of gravity (cog). Variable `valor_crisp` acquires such result.

4.3. On the construction of fuzzy sets

Since the actual fuzzy model, and the corresponding fuzzy rules base is entirely supported on fuzzy sets, it is important to show how JFK employs the object-oriented advantages to create them. The Java code of class `conjunto_difuso` is given in (Fig. 4). Class `conjunto_difuso.java` instantiates four objects, all of which refer to different types of membership functions. At this regard, classes `Triangular`, `Trapezoidal`, `Gaussiana` and `Sigmoidal` are called so that a particular membership function is generated. Through method overloading, two ways of calculating a membership function are provided. See, for example, the following two methods:

```
double[]    triangular(int uodi, int uodf,
double a, double  b, double c)
double triangular(double punto, double a,
double b, double c)
```

By calling the first method, an entire fuzzy set is obtained, over a given UoD. The membership function is stored as an array of double values. This is achieved through a constructor that receives five parameters. Two of them are the initial and final values of the UoD, and the remaining three are specific to the triangular function. As can be seen in Fig. 4, the returned value is a variable called `conjunto_generado`, which is a array of double values representing the computed fuzzy set. However, if the second method is called, the value obtained is the membership function for a specific crisp input value. In this case it not necessary to generate the entire fuzzy set. The computed value is stored in a variable called `valor_membresia`. This constructor is used in the antecedent clauses of the Mamdani rule. To exemplify and validate the proposed model, an application case is presented next.

5. On the use of JFK

The ranking of swimmers is the problem to illustrate the usage of JFK. Swimmers must record their specific times in several distances D (50, 100, 200, 400, 800 m.), styles STY (free style, backstroke, butterfly and breast), and two types of swimming pools P (short course (sc) and long course (lc)). According to registered times, swimmers are given points, which determine the swimmer's ranking. Current pointing system used by FINA is done with formula (8). This calculation has a severe limitation, because it only takes into account one input (the actual registered time for a given distance, style and piscine).

$$\text{Points} = 1000^{*}(\text{Base Time}/\text{Registered Time})^{3} \tag{8}$$

So far, there is no formula to calculate a swimmer's ranking by integrating multiple inputs. We propose to model this multiple input-one output system by means of fuzzy logic, and implement the resultant fuzzy rule base with JFK. An extract of the resultant fuzzy rule base is given in Eq. (9).

```
R1: IF t5msc is E5SC and t1msc is E1SC and t2msc is E2SC
and t4msc is E4SC and t8msc is E8SC THEN pointing is Good
R4: IF t5msc is P5SC and t1msc is P1SC and t2msc is PE2SC
and t4msc isP4SC and t8msc is P8SC THEN pointing is Good
```
(9)

Where t50frsc stands for the registered time in 50 meters, free style, on a short course piscine. E5SC stands for Excellent in a 50-m trial, on a short course piscine. P5SC stands for Poor in a 50-m trial, in a short court piscine. A similar notation is used for other distances, styles and piscines. In Fig. 5 it is shown how to instantiate objects of JFK to construct the swimmers rule base. Registered times are obtained by querying a database. The integration of JKF with Mysql was achieved, where crisp inputs are passed to JFK and the resultant crisp pointing is stored in the data base. The application is a prototype of a fuzzy decision sys-

```
Clausula_Mamdani te5sc = new Clausula_Mamdani("E5SC", E5SC, t5sc, 20.61, 20.61,
22.59, 23.11);

Clausula_Mamdani te1sc = new Clausula_Mamdani("E1SC", E1SC, t1sc, 45.33, 45.33, 49.2,
50.81);

Clausula_Mamdani te2sc = new Clausula_Mamdani("E2SC", E2SC, t2sc, 100.06, 100.06,
108.5, 112.17);

Clausula_Mamdani te4sc = new Clausula_Mamdani("E4SC", E4SC, t4sc, 212.31, 212.31,
230.24, 238.01);

Clausula_Mamdani te8sc = new Clausula_Mamdani("E8SC", E8SC, t8sc, 444.83, 444.83,
479.0, 498.65);


Clausula_Mamdani good = new Clausula_Mamdani("good", GOOD, puntaje_i, puntaje_f,
780.0, 850.0, 1100.0, 1100.0);

antecedentes_regla1 = new Clausula_Mamdani[]{te5sc, te1sc, te2sc, te4sc, te8sc};

regla_mamdani regla1 = new regla_mamdani(1, antecedentes_regla1, good);

reglas = new regla_mamdani[]
{
      regla1, regla2, regla3, regla4, regla5, regla6, regla7, regla8, regla9,
      regla10, regla11, regla12, regla13, regla14, regla15, regla16, regla17,
      regla18, regla19, regla20, regla21, regla22, regla23, regla24, regla25,
      regla26, regla27, regla28
};

brm = new Base_Reglas_Mamdani("natacion", reglas);
ranking = brm.resultado;
```

Fig. 5. Instantiating JFK objects.

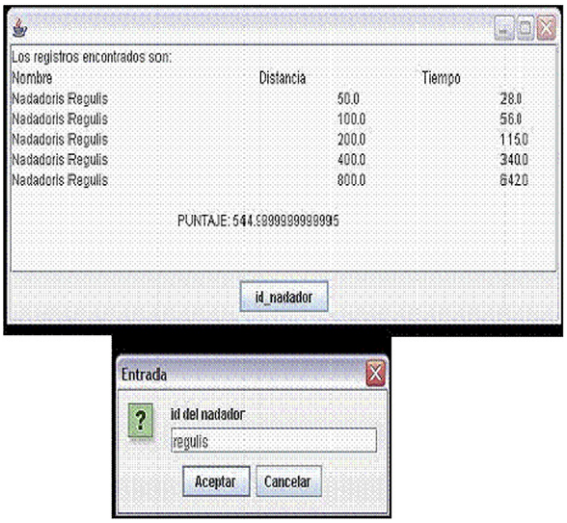

Fig. 6. Application of JFK to swimmers' ranking.

tem, which merges information and knowledge. Results are presented in Fig. 6.

6. Future work. Fuzzy distributed decision making

Future work is envisioned in two directions. One line of R + D deals with adding capabilities to JFK. The currently available version does not include objects to compute alpha cuts and levels. This feature is necessary to compute vagueness with more precision. The second line of research is based on applying JFK to distributed decision making under the Multi-Agent Systems (MAS) approach. At this regard, the swimmers problem is actually a metaphor for distributed decision making. To do so the well-known contract net protocol is extended, to select teams of swimmers (contractors) according to the importance of competition (the job offered). The importance of the competition is in turn evaluated by two input variables, the number of points and the monetary prize. The agent in charge of selecting and assigning swimmers to competitions possesses two different fuzzy expert systems. One of them, as explained lines above, determines the ranking of swimmers. Once this is completed, competitions are evaluated and a matching algorithm determines what swimmers must compete, in order to maximize profits. With this approach, the selection problem is facilitated, since classical approaches use linear programming methods, modified to incorporate vagueness. Example of this is (Kumar, Vrat, & Shankar, 2006). However, these approaches do not model the system by means of fuzzy expert systems, and their mathematical complexity remains.

Distributed fuzzy decision making is being accomplished by integrating JFK and the standardized platform known as JADE (Java Agent Development Environment). At this regard agents with fuzzy expert systems are being developed and tested, along with communication protocols. Newly developed ideas include dynamical membership functions, as can be found in (Cerrada, Aguilar, Colina, & Titli, 2005).

7. Conclusions

An Application Programming Interface for developing fuzzy expert systems is presented. The modelling complies with a general design pattern found in the structure of the Mamdani model. The dynamics of the system respond to the inference algorithm based on the generalized principle of extension. JFK has been tested with the swimmers' ranking problem as an example of a multi-variable decision system. We provide a smooth integration of information and knowledge, with the coordinated use of JFK and MySql. Future work includes the modelling and realization of fuzzy distributed decision making systems, by integrating JFK and the standardized platform Java Agent Development Environment (JADE).

References

Babuska, R. (1998). *Fuzzy modeling for control*. Boston, USA: Kluwer Academic Publishers.

Bigus, J., & Bigus, J. (2001). *Constructing intelligent agents using Java*. USA: John Wiley and Sons.

Cerrada, M., Aguilar, J., Colina, E., & Titli, A. (2005). Dynamical membership functions: an approach to adaptive fuzzy modelling. *Fuzzy Sets And Systems, 152*, 513–533.

Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language* (3rd ed.). Boston: Addison–Wesley.

Friedman-Hill, E. (2003). *Jess in action: Java rule based systems*. USA: Manning Publications Co.

Gamma, E. et al. (1995). *Design patterns. Elements of reusable object-oriented software*. CA: Addisson–Wesley.

Horng, M.-F., Lee, S.-C., & Kuo, Y.-H. (2003). Object-oriented framework of fuzzy knowledge systems. *Studies In Fuzziness And Soft Computing, 121*, 171–182.

Jan, R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. USA: Prentice-Hall.

Juang, Y.-S., Lin, S.-S., & Kao, H.-P. (2007). Design and implementation of a fuzzy inference system for supporting customer requirements. *Expert Systems With Applications, 32*(3), 868–878.

Kumar, M., Vrat, P., & Shankar, R. (2006). A fuzzy programming approach for vendor selection problem in a supply chain. *International Journal of Production Economics, 101*, 273–285.

Ling, Y.-F., Tseng, S. S., & Tsai, C.-F. (2003). Design and implementation of a new object-oriented rule-based management system. *Expert Systems with Applications, 25*, 369–385.

Orchard, R. (2006). National Research Council of Canada's Institute for Information Technology, Canada, URL: http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html.

Perfilieva, I. (2006). Logical foundations of rule-based systems. *Fuzzy Sets And Systems, 157*, 615–621.

Sendelj, R., & Devedzic, V. (2004). Fuzzy systems based on component software. *Fuzzy Sets And Systems, 141*(3), 487–504.

Shu-Hsien, L. (2005). Expert system methodologies and applications. A decade review from 1995 to 2004. *Expert Systems With Applications, 28*(1), 93–103.

Souza, M. A. F., & Ferreira, M. A. G. V. (2002). Designing reusable rule-based architectures with design patterns. *Expert Systems With Applications, 23*, 395–403.

Stoll, R. R. (1979). *Set theory and logic*. New York: Dover.

Wong, G. Y. C., & Chun, H. W. (1999). Modeling fuzzy sets using object-oriented techniques. *Lecture Notes In Computer Science, 1611*, 23–32.

Yen, J., & Langari, R. (1999). *Fuzzy logic: Intelligence, control and information*. USA: Prentice-Hall.

Zadeh, L. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions On Systems*, Man and Cybernetics, SMC-3.