
Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro ISYS, Laboratorio de Inteligencia Artificial

# Prototipo de herramienta para el desarrollo de aplicaciones basadas en lógica difusa

Trabajo Especial de Grado presentado ante la ilustre
**Universidad Central de Venezuela**
por la bachiller
**Patricia O'Callaghan Olivo**
Para optar al título de
**Licenciada en Computación**

**Tutores**
Profa. Haydemar Núñez
Prof. Iván Flores

Caracas, Abril del 2011

# PROTOTIPO DE HERRAMIENTA PARA EL DESARROLLO DE APLICACIONES BASADAS EN LÓGICA DIFUSA

## Resumen

La creciente popularidad de la lógica difusa se debe a su buena adaptación a problemas donde existen valores numéricos difíciles de medir, incertidumbre, o excesiva complejidad, así como también a su capacidad de resolver problemas de forma potente y precisa y a la variedad de sus aplicaciones. Actualmente, existen numerosas herramientas para el desarrollo de aplicaciones basadas en lógica difusa, sin embargo, aquellas que cuentan con un mayor número de funcionalidades y mejor usabilidad no son de libre distribución.

En este Trabajo Especial de Grado se presenta un prototipo de herramienta que cumple con las funcionalidades necesarias para el desarrollo de sistemas basados en reglas difusas, bajo un ambiente gráfico que facilita el trabajo del usuario, con una interfaz agradable e intuitiva que permite guiarlo a través de todos los pasos necesarios para desarrollar su aplicación, además de ser de libre distribución.

**Palabras clave:** Lógica difusa, Sistemas basados en reglas difusas, variables lingüísticas, conjuntos difusos, reglas difusas, inferencia difusa, aplicaciones.

# CONTENIDO

# INDICE DE TABLAS

# INDICE DE FIGURAS

# INTRODUCCIÓN

La lógica difusa brinda una base matemática para poder modelar conceptos imprecisos que se utilizan en el lenguaje diario y que no pueden ser definidos de manera exacta. Esto permite resolver un gran número de problemas que no podrían conseguir una solución de manera satisfactoria dada la naturaleza del dominio donde se presentan y los parámetros que los rodean.

Desde 1965, año en el cual se publicó el artículo "*Fuzzy Sets*" (Zadeh, 1965), el uso de la lógica difusa comenzó a popularizarse. Actualmente, la implementación exitosa de herramientas basadas en lógica difusa ha traído como consecuencia un incremento de su utilización en aplicaciones para sistemas de control industriales, reconocimiento de patrones, sistemas expertos, electrodomésticos, minería de datos, etc.

Para el diseño de sistemas basados en reglas difusas existe una amplia gama de herramientas disponibles, sin embargo, la mayoría no son de libre distribución, o su usabilidad es limitada. Por esta razón, el objetivo principal de este trabajo es desarrollar un prototipo de herramienta para el desarrollo de sistemas basados en reglas difusas, el cual permita a los usuarios contar con un ambiente de desarrollo potente y fácil de utilizar y que adicionalmente sea de distribución libre.

Este documento se estructura en dos (2) capítulos. El primero, desarrolla el marco teórico de la investigación, dentro del cual se describe todo lo referente a la teoría de conjuntos difusos, el desarrollo de sistemas basados en reglas difusas y sus principales aplicaciones. También se hace una revisión de herramientas ya existentes para la resolución de problemas con lógica difusa. Finalmente, en el segundo capítulo se expone el marco aplicativo, donde se plantea el problema que se analiza en este Trabajo Especial de Grado, así como sus objetivos y se indican los aspectos referentes al desarrollo del prototipo tales como análisis de los requerimientos, modelos de casos de uso, la arquitectura del sistema y el diseño de la interfaz gráfica. Por último, se muestran los experimentos hechos sobre la herramienta y sus resultados.

# CAPÍTULO I: MARCO TEÓRICO

En este capítulo se presentará todo lo referente al marco teórico de la lógica difusa: introducción, conjuntos difusos, relaciones difusas, reglas difusas, razonamiento aproximado e inferencia difusa, desarrollo de sistemas difusos y revisión de herramientas para lógica difusa.

## 1. 1 Introducción a la lógica difusa

A diario expresamos ideas apoyándonos en nuestro lenguaje, utilizando comúnmente frases como “Tomás es alto”, en lugar de “Tomás mide 1.84 metros”; sin embargo, la percepción de “alto” es relativa, ya que una persona puede catalogar a un individuo como “alto” si tiene una estatura mayor a 1.80 metros y para otra lo puede ser sobre los 1.70 metros. Este tipo de sentencias dependen entonces del contexto sobre el cual son evaluadas.

En la teoría clásica de conjuntos, se podrían definir los elementos pertenecientes al conjunto “alto” como todos aquellos individuos con estatura mayor o igual a 1.80 metros. Bajo esta definición, ¿no es acaso impreciso determinar que un individuo con altura igual a 1.79 metros no debe ser considerado como perteneciente al conjunto “alto” y que debe ser asignado al conjunto “mediano” junto con un individuo con altura igual a 1,60 metros?

En vista de que la teoría clásica de conjuntos no puede manejar este tipo de imprecisiones, fue entonces ideada la teoría de los conjuntos difusos, permitiendo que los elementos puedan pertenecer con diferentes grados a distintos conjuntos difusos. Por ejemplo, un individuo con altura igual a 1.60 metros posee un grado de pertenencia al conjunto difuso “medio” mucho mayor que un individuo de 1.89 metros, el cual pertenece al conjunto difuso en un grado muy cercano a cero.

Históricamente, la teoría de los conjuntos difusos tiene sus bases iniciales en los planteamientos hechos por Lotfi A. Zadeh de la Universidad de California-Berkeley, en su artículo publicado en 1965 llamado “*Fuzzy Sets*” (Zadeh, 1965), donde desarrolla una extensión de los conjuntos clásicos, llamados “Conjuntos Difusos” y define las operaciones aplicables a estos.

Inicialmente, Zadeh ideó la lógica difusa para poder representar y manipular eficientemente la imprecisión y vaguedad del razonamiento humano expresado lingüísticamente, sin embargo, las primeras aplicaciones de la lógica difusa fueron utilizadas en el control de procesos, a principio de la década de los 80.

Desde entonces, la lógica difusa ha sido aplicada para diversas áreas como la minería de datos, el reconocimiento de patrones, sistemas de predicción, sistemas de diagnósticos médicos, captura del movimiento, etc.

En la actualidad, la lógica difusa es aplicada a nuevos campos, junto con la utilización de las redes neuronales, el razonamiento probabilístico y los algoritmos genéticos, entre otros.

La creciente popularidad de la lógica difusa se debe a su buena adaptación a problemas donde existen valores numéricos difíciles de medir, indeterminación o incertidumbre, o excesiva complejidad, lo que trae como consecuencia el desconocimiento parcial o total del modelo matemático de dichos sistemas.

### 1.1.1 Conjuntos difusos

La teoría de los conjuntos difusos es una extensión de la teoría clásica de conjuntos. Se tiene como definición de un conjunto difuso la siguiente (Nguyen y Walker, 1996):

Sea $U$ el universo de discurso y $x$ un elemento perteneciente a $U$. Un conjunto difuso $A$ de $U$ está caracterizado por la función de pertenencia $\mu_A : U \rightarrow [0,1]$, la cual asocia cada elemento $x \in U$ con un numero $\mu_A(x)$ que representa el grado de pertenencia de $x$ en $A$, y está designado como:

$$A = \{ (x, \mu_A(x)) \mid x \in U \} \tag{1.1}$$

Por ejemplo, si se tiene un conjunto difuso etiquetado “JOVEN” cuyos elementos son aquellas personas consideradas jóvenes y cuyos grados de pertenencia dependen de sus edades, se puede decir que una persona de 72 años de edad tiene un grado de pertenencia cercano a 0 y una persona con 2 años de edad tiene un grado de pertenencia igual a 1. Las personas con edades comprendidas entre 2 y 72 años tienen grados de pertenencia con valores entre 0 y 1. Debido a que es subjetivo el considerar si una persona es joven o no, existen muchas representaciones para el conjunto “JOVEN”, observándose una de ellas en la Tabla 1.1:

*Tabla 1.1. Representación del conjunto difuso "JOVEN"*

| JOVEN | |
|---|---|
| **Edad** | **Función de pertenencia** |
| ≤25 | $\mu_A(x \leq 25) = 1$ |
| 30 | $\mu_A(30) = 0.8$ |
| 35 | $\mu_A(35) = 0.6$ |
| 40 | $\mu_A(40) = 0.4$ |
| 45 | $\mu_A(45) = 0.2$ |
| ≥50 | $\mu_A(x \geq 50) = 0$ |

La representación gráfica de este conjunto se observa en la siguiente Figura (Figura 1.1):

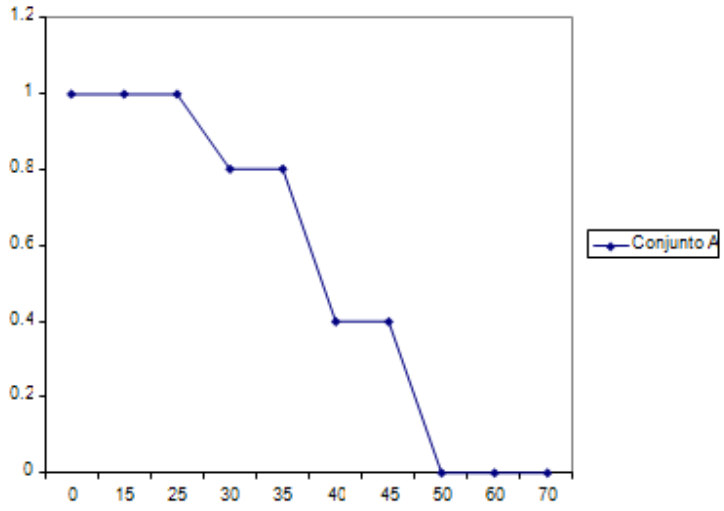
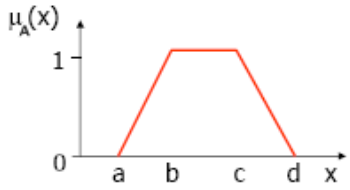
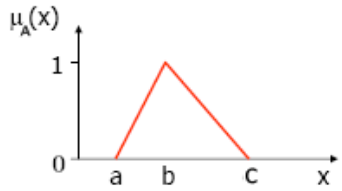
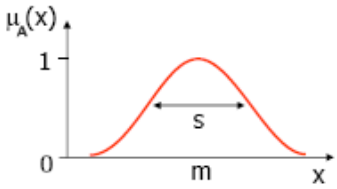



*Figura 1.1. Representación gráfica del conjunto difuso "JOVEN"*

Para definir conjuntos difusos, se deben determinar las funciones de pertenencias asociadas, las cuales toman valores de forma continua entre 0 y 1. Para determinar la función de pertenencia asociada a un conjunto, se puede recolectar la información a partir del conocimiento de expertos o de una colección de datos. Existen una serie de funciones de pertenencia que se utilizan comúnmente, algunas de las cuales se describen en la Tabla 1.2, según se indica en (Nguyen y Walker, 1996):

*Tabla 1.2. Descripción de las principales funciones de pertenencia*

| Función | Descripción |
|---|---|
| **Trapezoidal** | Descripción |
| | Tiene como parámetros los valores a, b, c, d y cumple que a<b<c<d. |
| | Definición |
| | $$\mu_A(x) = \begin{cases} 0 & \text{si } (x>a) \text{ ó } (x>d) \\ (x-a)/(b-a) & \text{si } a \leq x \leq b \\ 1 & \text{si } b \leq x \leq c \\ (d-x)/(d-c) & \text{si } c \leq x \leq d \end{cases}$$ |
| | Gráfica |
| | Trapezoidal: <a,b,c,d><br>  <br> |
| **Triangular** | Descripción |
| | Tiene como parámetros los valores a, b, c y cumple que a<b<c. |
| | Definición |
| | $$\mu_A(x) = \begin{cases} 0 & \text{si } (x<a) \\ (x-a)/(b-a) & \text{si } a < x \leq b \\ (c-x)/(c-b) & \text{si } b < x < c \\ 0 & \text{si } x \geq c \end{cases}$$ |

| | |
|---|---|
| | Gráfica |
| | Triangular: <a,b,c><br>  <br> |
| **Gaussiana** | Descripción |
| | Se define por su valor medio m y una desviación estándar s > 0. Se cumple que cuanto menor es s, más estrecha es la campana. |
| | Definición |
| | $\mu_A(x) = \ell^{-((x-m)^2)/(2(s^2))}$ |
| | Gráfica |
| | Gaussiana: N(m,s)<br>  <br> |

También es importante reseñar algunos términos básicos asociados a la teoría de conjuntos difusos como lo son soporte, conjunto difuso unitario, corte-α, altura, conjunto difuso normal, núcleo y punto de cruce (Nguyen y Walker, 1996):

- Soporte: se define como el conjunto clásico $A$ de elementos cuyo grado de pertenencia es mayor que cero.

$$\text{Soporte}(A) = \{\ x \in U \,/\, \mu_A(x) > 0\ \} \qquad (1.2)$$

Un conjunto difuso es *unitario* si su soporte está formado por un único elemento.

En el ejemplo descrito en la Tabla 1.1, el conjunto soporte lo conforman todos los elementos del conjunto difuso “JOVEN” con valores mayores a 50.

- Corte-α: se define como el conjunto clásico $A$ de elementos cuyo grado de pertenencia es mayor o igual a α. También se define el corte-α estricto, que es representado por el conjunto clásico $A$ de elementos cuyo grado de pertenencia es estrictamente mayor a α.

$$\text{corte-}\alpha(A) = \{\ x \in U \,/\, \mu_A(x) >= \alpha\ \} \tag{1.3}$$

$$\text{corte-}\alpha\text{ estricto}(A) = \{\ x \in U \,/\, \mu_A(x) > \alpha\ \} \tag{1.4}$$

  Por ejemplo, si se toma un valor α igual a 0.6, el conjunto corte-α asociado al ejemplo ilustrado en la Tabla 1.1, estaría conformado por todos aquellos elementos del conjunto difuso “JOVEN” con valores iguales o menores a 35.

- Altura: se define por el valor máximo de la función de pertenencia de un conjunto difuso $A$. Un conjunto difuso es *normal* si su altura es igual a 1.

  En el ejemplo ilustrado en la Tabla 1.1, se puede observar que el conjunto difuso “JOVEN” es normal ya que para todos los elementos con valores igual o menores a 25 la función de pertenencia retorna un valor igual a 1.

- Núcleo: consiste en el conjunto clásico $A$ de elementos que tienen grado de pertenencia 1.

$$\text{Núcleo}(A) = \{\ x \in U \,/\, \mu_A(x) = 1\ \} \tag{1.5}$$

  En el ejemplo ilustrado en la Tabla 1.1, se puede observar que el núcleo del conjunto difuso “JOVEN” está compuesto por todos los elementos con valores igual o menores a 25.

- Punto de cruce: es el elemento $x$ para el cual la función de pertenencia es igual a 0.5.

  En el ejemplo ilustrado en la Tabla 1.1, se puede observar que el punto de cruce es el elemento del conjunto difuso “JOVEN” con valor igual a 37.5, ya que la función de pertenencia asociada a este elemento retorna un valor igual a 0.5.

A continuación se describirán las principales operaciones básicas que se pueden aplicar a los conjuntos difusos:

Complemento: Sea un conjunto difuso $A$, su complemento es conocido como conjunto difuso $A$', y es definido por:

$$\mu_{A'}(x) = C(\mu_A), \ \forall \ x \in U \tag{1.6}$$

donde $C(\mu_A)$ es una *C-norma*, una función que cumple con las siguientes propiedades:

- Contorno:

$$C(0) = 1, C(1) = 0 \tag{1.7}$$

- No incremento:

$$\text{si } (x < y) \Rightarrow C(x) > C(y) \tag{1.8}$$

$$\forall \ x, y \in [0,1]$$

- Involución:

$$C(C(x)) = x \tag{1.9}$$

$$\forall \ x, y \in [0,1]$$

Algunas funciones que cumplen con estas propiedades y que son comúnmente utilizadas para la operación de complemento son:

- Negación: $C(x) = 1 - x$ (1.10)
- Sugeno: $C(x) = (1-x)/(1+\lambda)$ ; con $\lambda > 0$ (1.11)
- Yager: $C(x) = (1-x^w)^{1/w}$ ; con $w > 0$ (1.12)

Por ser la más utilizada, sólo se ilustrará la operación de complemento con la función Negación. Se hace referencia al conjunto difuso $A$ "JOVEN", el cual fue definido anteriormente. El conjunto complementario $A$' podría ser etiquetado como "VIEJO". Se ilustra en la Figura 1.2:

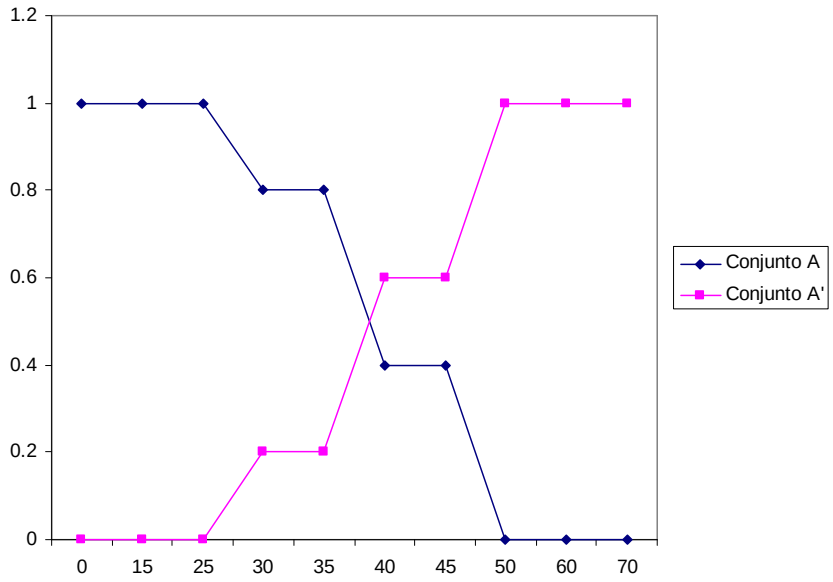



*Figura 1.2. Conjunto A y su complemento A'*

Intersección: En la teoría clásica de conjuntos, un elemento pertenece al conjunto intersección de dos conjuntos si pertenece a ambos; en la teoría de conjuntos difusos lo que se determina es el grado de pertenencia de dicho elemento al conjunto intersección de dos conjuntos, conociendo el grado de pertenencia del elemento a cada uno de los conjuntos.

Sean los conjuntos difusos $A$ y $B$ definidos sobre $U$, se define la intersección de ambos conjuntos $A \cap B$ cuya función de pertenencia es determinada por:

$$\mu_{A \cap B}(x) = T(\mu_A, \mu_B), \ \forall \ x \in U \qquad (1.13)$$

donde $T(x, y)$ es una *T-norma*, una función que cumple con las siguientes propiedades:

- Conmutatividad: $T(x, y) = T(y, x), \ \forall \ x, y \in U$ (1.14)
- Asociatividad: $T(x, T(y, z)) = T(T(x, y), z), \ \forall \ x, y, z \in U$ (1.15)
- Monotonía:
  si $(x \leq y) \ \wedge \ (w \leq z) \ \Rightarrow \ T(x, w) \ \leq \ T(y, z) \ , \ \forall \ x, y, z, w \in U$
  (1.16)
- Elemento absorbente: $T(x, 0) = 0, \ \forall \ x \in U$ (1.17)
- Elemento neutro: $T(x, 1) = x, \ \forall \ x \in U$ (1.18)

Algunas funciones que cumplen con estas propiedades y que son comúnmente utilizadas para la operación de intersección son:

- Mínimo: $T(x, y) = \min(x, y)$ (1.19)
- Producto algebraico: $T(x, y) = x \cdot y$ (1.20)
- Producto drástico: $T(x, y) = \begin{cases} x & \text{si } y = 1 \\ y & \text{si } x = 1 \\ 0 & \text{en caso contrario} \end{cases}$ (1.21)

Por ser la más utilizada, sólo se ilustrará la operación de intersección con la función Mínimo. Se hace referencia al conjunto difuso $A$ "JOVEN", el cual fue definido anteriormente, y al conjunto difuso $B$ "VIEJO". En el siguiente gráfico (Figura 1.3) se puede observar la intersección de dos conjuntos difusos.

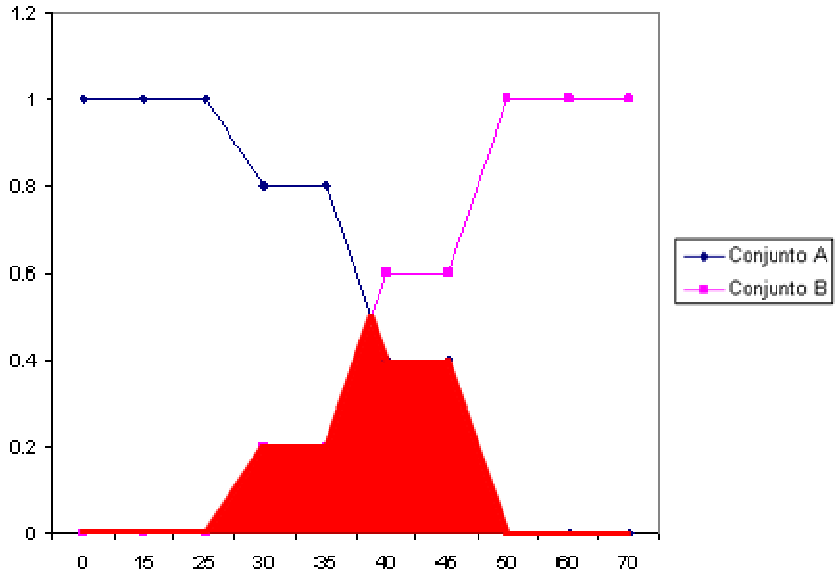



*Figura 1.3. Intersección de 2 conjuntos difusos*

Unión: Sean los conjuntos difusos $A$ y $B$ definidos sobre $U$, se define la unión de ambos conjuntos $A \cup B$ cuya función de pertenencia es determinada por:

$$\mu_{A \cup B}(x) = S(\mu_A, \mu_B), \ \forall \ x \in U \qquad (1.22)$$

donde $S(\mu_A, \mu_B)$ es una *T-conorma*, una función que cumple con las siguientes propiedades:

- Conmutatividad: $S(x, y) = S(y, x),\ \forall\ x, y \in U$ (1.23)
- Asociatividad: $S(x, S(y, z)) = S(S(x, y), z),\ \forall\ x, y, z \in U$ (1.24)
- Monotonía:
  si $(x \leq y) \wedge (w \leq z) \Rightarrow S(x, w) \leq S(y, z),\ \forall\ x, y, z, w \in U$ (1.25)
- Elemento absorbente: $S(x, 0) = 0,\ \forall\ x \in U$ (1.26)
- Elemento neutro: $S(x, 1) = x,\ \forall\ x \in U$ (1.27)

Algunas funciones que cumplen con estas propiedades y que son comúnmente utilizadas para la operación de unión son:

- Máximo: $S(x, y) = \max(x, y)$ (1.28)
- Suma algebraica: $S(x, y) = x + y - x \cdot y$ (1.29)
- Suma drástica: $S(x, y) = \begin{cases} x & \text{si } y = 0 \\ y & \text{si } x = 0 \\ 1 & \text{en caso contrario} \end{cases}$ (1.30)

Por ser la más utilizada, sólo se ilustrará la operación de unión con la función Máximo. Se hace referencia al conjunto difuso $A$ "JOVEN" y al conjunto difuso $B$ "VIEJO". En el siguiente gráfico (Figura 1.4) se puede observar la unión de dos conjuntos difusos.

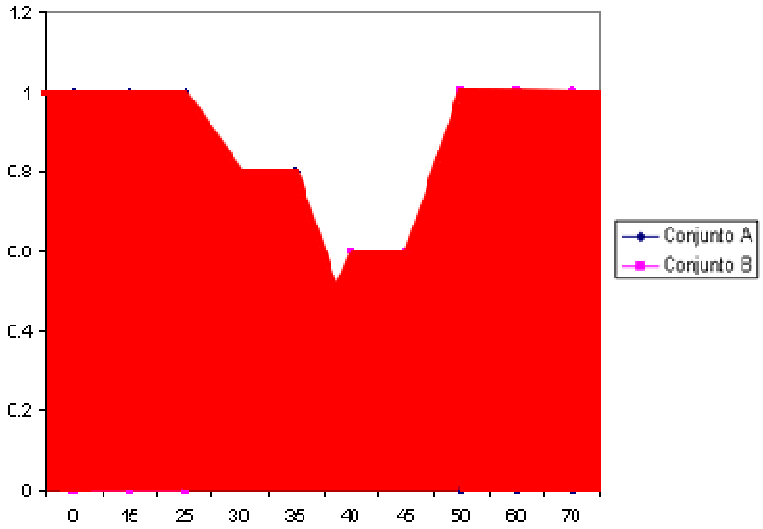



*Figura 1.4. Unión de 2 conjuntos difusos*

Es importante destacar que existen dos leyes fundamentales de la teoría de conjuntos clásica que no pueden extenderse a la teoría de conjuntos difusos, éstas son el principio del tercero excluido, la cual indica que la unión entre un conjunto $A$ y su complemento $A$' es igual al universo de discurso $U$ ; y la ley de contradicción, la cual indica que la intersección de un conjunto $A$ y su complemento $A$' es igual al vacío. Por lo tanto, para los conjuntos difusos se cumple lo siguiente:

$$A \cup A' \neq U \tag{1.31}$$

$$A \cap A' \neq \emptyset \tag{1.32}$$

Los conjuntos difusos son definidos para caracterizar variables llamadas "variables lingüísticas". Estas variables toman valores que no son números sino palabras o sentencias en lenguaje natural o artificial (Zadeh, 1965); admiten valores conocidos como etiquetas lingüísticas, donde cada etiqueta lingüística es un conjunto difuso sobre un dominio subyacente (dominio numérico).

Sea $u$ la variable lingüística y sean $x$ los valores numéricos que se asignan a $u$, donde $x \in U$. Una variable lingüística es usualmente descompuesta en un conjunto de términos $T(u)$, el cual cubre su universo de discurso.

Por ejemplo, sea temperatura $(u)$ una variable lingüística. Puede ser descompuesta en el siguiente conjunto de términos. $T$ (temperatura) = {muy fría, fría, tibia, caliente, muy caliente}, donde cada término es caracterizado por un conjunto difuso en el universo de discurso $X$ =[-15 C, 45 C]. En el siguiente gráfico (Figura 1.5) se puede observar como son caracterizados cada uno de estos términos mediante conjuntos difusos:

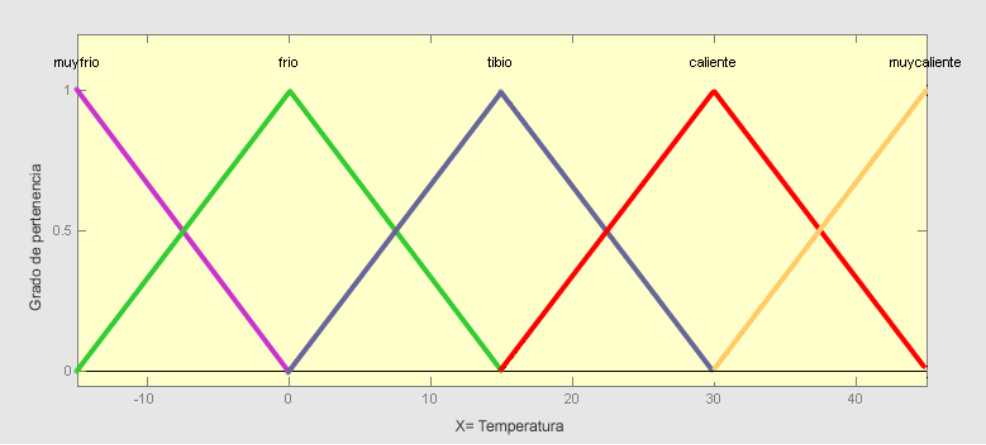



*Figura 1.5. Conjuntos difusos para el universo de discurso Temperatura*

Las variables lingüísticas son útiles ya que permiten trasladar conceptos lingüísticos a descripciones numéricas.

Como definición formal de una variable lingüística se tiene que es un conjunto de 5 elementos:

$$< N, Y, T(N), G, M >$$

Donde:

- $N$ es el nombre de la variable

- $U$ es el dominio subyacente

- $T(N)$ es el conjunto de términos o etiquetas que puede tomar la variable

- $G$ es la gramática para generar las etiquetas de $T(N)$, por ejemplo “muy frío”, “frío”, “tibio”, “caliente”, “muy caliente”. Los símbolos terminales de la gramática incluyen un conjunto de términos primarios (“bajo”, “alto”, etc), un conjunto de modificadores (“muy”, “cercano a”, “más”, “menos”, etc) y un conjunto de conectivos lógicos (NO, Y, O).

- $M$ es una regla semántica que asocia cada elemento de $T(N)$ con un conjunto difuso en $U$ de entre todos los posibles:

$$M : T(N) \rightarrow F(U)$$

## 1.1.2 Relaciones difusas

Las relaciones difusas representan un grado de asociación entre los elementos de 2 o más conjuntos difusos (Li, 2006). Sean $U$ y $V$ dos universos de discurso, se establece la relación difusa $R(U, V)$ la cual es un conjunto difuso en el espacio producto $U \times V$ que se caracteriza por la función de pertenencia $\mu_R(x, y)$, la cual indica el grado de relación entre $x$ y $y$, donde $x \in U$, $y \in V$, es decir

$$R(U, V) = \{ ((x, y), \mu_R(x, y)) | (x, y) \in U \times V \} \quad (1.33)$$

En la Tabla 1.3 se representa la relación “a es similar a b”:

*Tabla 1.3. Relación “a es similar a b”*

| $V \backslash U$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 0.7 | 0.3 | 0 |
| 1 | 0.7 | 1 | 0.7 | 0.3 |
| 2 | 0.3 | 0.7 | 1 | 0.7 |
| 3 | 0 | 0.3 | 0.7 | 1 |

Las operaciones difusas y operadores definidos anteriormente pueden ser aplicadas a las relaciones difusas, al ser éstas un conjunto difuso en el espacio producto.

Sean $R(x, y)$ y $S(x, y)$ dos relaciones en el mismo espacio producto $U$ x$V$ :

Intersección: La intersección entre $R$ y $S$ sería definida de la siguiente forma:

$$\mu_{R \cap S}(x, y) = \mu_R(x, y) * \mu_S(x, y) \tag{1.34}$$

donde * es cualquier T-norma

Unión: La unión entre $R$ y $S$ sería definida de la siguiente forma:

$$\mu_{R \cup S}(x, y) = \mu_R(x, y) \oplus \mu_S(x, y) \tag{1.35}$$

donde $\oplus$ es cualquier T-conorma

Sean $R(x, y)$ y $S(x, y)$ dos relaciones que pertenecen a diferentes espacios producto $R$ ( $U$ ,$V$ ) y $S$ ($V$ ,$W$ ):

Composición: La composición difusa entre $R$ y $S$ , se define como una relación difusa en $U$ x$W$ cuya función de pertenencia viene dada por:

$$\mu_{R \circ S}(x, z) = \sup_{y \in V} [\mu_R(x, y) * \mu_S(y, z)] \tag{1.36}$$

Donde el operador sup es el máximo y el operador * puede ser cualquier T-norma. Dependiendo de la T-norma seleccionada se pueden obtener distintas composiciones, siendo las más usadas la max-min y la max-product.

## 1.1.3 Reglas difusas

El razonamiento difuso es realizado a partir de reglas de inferencia difusas.

Según (Nguyen y Walker, 1996), una regla SI – ENTONCES difusa es una sentencia condicional de la siguiente forma:

$$R^{t}_{1} = \text{“SI”}\ x_1\ \text{”ES”}\ F^{t}_{1}\ \text{“Y” ... “Y” “SI”}\ x_n\ \text{”ES”}\ F^{t}_{n}\ \text{”ENTONCES”}\ y\ \text{“ES”}\ G^{t} \quad (1.37)$$

donde

$F^{t}$ y $G^{t}$ , son conjuntos difusos

$x = (x_1, \ldots, x_n)^T \in U$ , son las variables lingüísticas de entrada

$y \in V$ , es la variable lingüística de salida

Existen dos tipos de sentencias difusas:

- Atómicas: Por ejemplo, “temperatura ES caliente”
- Compuestas: Es una colección de sentencias difusas unidas por operadores lógicos como Y, O, NO. Por ejemplo, “temperatura ES caliente Y humedad ES alta”

Como ejemplo de una regla difusa se tiene:

SI “temperatura” ES “caliente” Y “humedad” ES “alta” ENTONCES

“temperatura aire acondicionado” ES “alta”

## 1.1.4 Razonamiento aproximado e inferencia difusa

Según Zadeh (1979), “*informalmente, por razonamiento aproximado o, equivalentemente, razonamiento difuso, se entiende como el proceso o procesos mediante los que una conclusión imprecisa se deduce de una colección de premisas imprecisas. Tal forma de razonamiento es en su mayor parte más cualitativo que cuantitativo en su naturaleza y casi todo él cae fuera del dominio de aplicación de la lógica clásica*”.

En la lógica difusa se utiliza el razonamiento aproximado, el cual permite obtener conclusiones difusas a partir de premisas difusas basándose en la teoría de conjuntos difusos. Entonces, al extender la clásica regla “Modus Ponens” a este tipo de razonamiento, se tiene la regla de inferencia difusa llamada “Modus Ponens aproximado”.

El “Modus Ponens aproximado” se puede representar de la siguiente manera:

*Regla*: Si *x* es A entonces *y* es B

*Sentencia*: *x* es A’

---

*Conclusión*: *y* es B’

Donde A, B, A’ y B’ son conjuntos difusos, A’ y B’ son parecidos a A y B respectivamente (pero no idénticos) (Bonissone, 1999).

Por ejemplo, se tiene la regla “Si *x* es pequeño entonces *y* es grande” y la sentencia “*x* es muy pequeño”, aplicando el “Modus Ponens aproximado” se tendría como conclusión una sentencia como “*y* es muy grande”.

Para determinar la expresión concreta del conjunto B’, dados A, B y A’, se cuenta con una regla que devuelve la función de pertenencia $\mu_{B'}(y)$ en función de $\mu_{A'}(x)$,$\mu_A(x)$ y $\mu_B(y)$, la cual es conocida como la regla composicional de inferencia y se describe a continuación (Nguyen y Walker, 1996):

$$B' = A' \circ (A \rightarrow B) \Rightarrow \mu_{B'}(y) = \max [\min (\mu_{A'}(x), \mu_{A \rightarrow B}(x, y))] \qquad (1.38)$$

Es importante recordar que la regla “Si *x* es A entonces *y* es B” puede ser considerada como una relación difusa en $X \times Y$ y que el conjunto difuso de salida B’ puede ser considerado como la imagen del conjunto difuso A’ a través de la relación de implicación.

Generalmente se utilizan T-normas como funciones de implicación y las más utilizadas son la implicación de Mamdani y la implicación de Larsen. La implicación de Mamdani se caracteriza por truncar el conjunto difuso de salida, mientras que la implicación de Larsen lo escala (Serrano, 2009).

- Implicación de Mamdani:

A continuación se presenta su definición y un ejemplo gráfico (Figura 1.6):

$$\mu_{A \rightarrow B}(x, y) = \min(\mu_A(x), \mu_B(y)) \qquad (1.39)$$

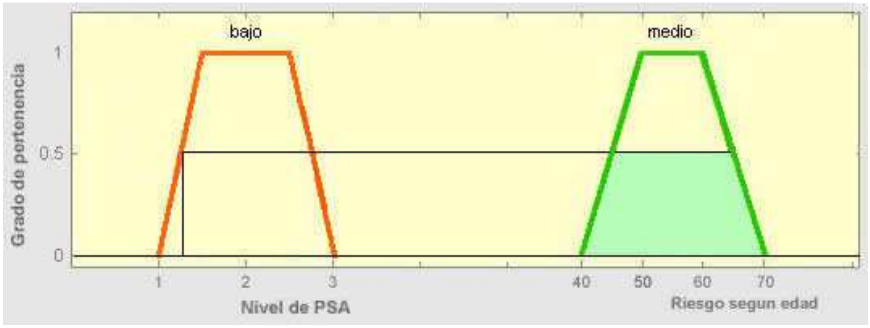



*Figura 1.6. Implicación de Mamdani.*

- Implicación de Larsen:

A continuación se presenta su definición y un ejemplo gráfico (Figura 1.7):

$$\mu_{A \to B}(x, y) = \text{prod}(\mu_A(x), \mu_B(y)) \tag{1.40}$$

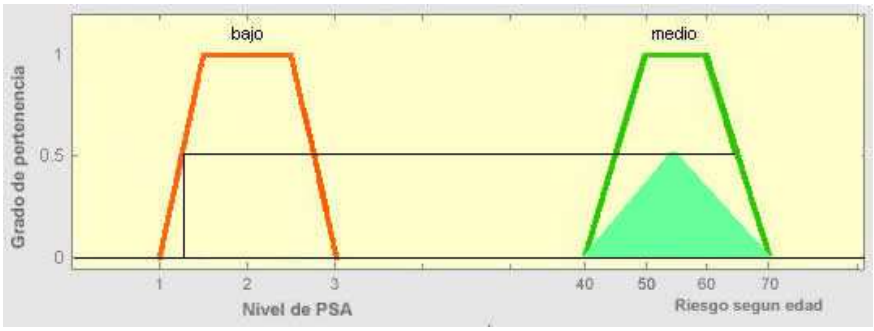



*Figura 1.7. Implicación de Larsen.*

La ejecución de cada regla se hará aplicando el método de inferencia seleccionado y el resultado será el conjunto difuso resultante de la implicación.

Es importante destacar que las reglas difusas pueden ser activadas en paralelo; además el orden en que son activadas no afecta el resultado. Por otra parte, a cada regla se le puede asignar un peso, el cual es aplicado al valor asociado al antecedente.

El grado de implicación es el resultado de aplicar operadores lógicos en el antecedente. El valor de entrada para el proceso de implicación es un número dado por el antecedente y la salida es un conjunto difuso.

Por último se realiza el proceso de agregación, el cual acumula todos los conjuntos difusos de salida asociados a cada regla activada durante el proceso de inferencia.

### 1.1.5 Desarrollo de sistemas difusos

Los sistemas de inferencia difusa interpretan los valores de un vector de entrada y realizando inferencias sobre un conjunto de reglas difusas, se asignan valores a un vector de salida. Este proceso se apoya en funciones de pertenencia, operaciones difusas, reglas difusas y agregación de los conjuntos de salida.

El siguiente esquema (Figura 1.8) ejemplifica un sistema de inferencia difusa:

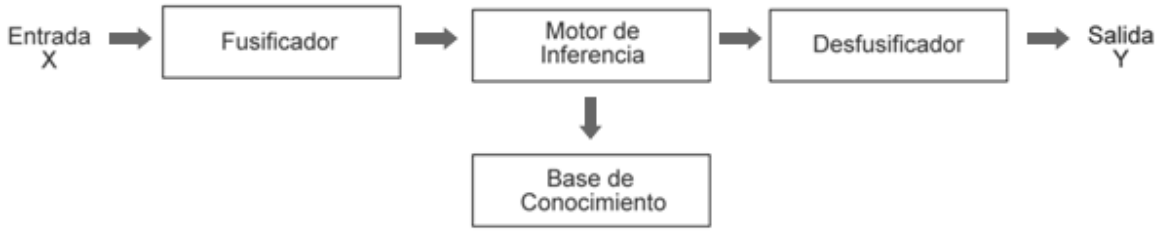



*Figura 1.8. Sistema de inferencia difusa*

La base de conocimientos contiene reglas lingüísticas que son generalmente proporcionadas por expertos; sin embargo, también es posible extraer reglas a partir de datos numéricos.

El fusificador toma los valores de entrada y determina el grado de pertenencia a cada uno de los conjuntos difusos a través de las funciones de pertenencia.

El motor de inferencia determina el grado en el cual el antecedente satisface a cada regla. Si el antecedente de una regla dada tiene más de una sentencia, los operadores difusos son aplicados para obtener un número que represente el resultado del antecedente para esa regla. Es importante recordar que es posible que una o más reglas se puedan activar al mismo tiempo.

Los conjuntos difusos de salida asociados a cada regla son luego acumulados durante el proceso de agregación, siendo combinados en un solo conjunto difuso. La salida del proceso de agregación es un conjunto difuso para cada variable de salida.

Existen varios métodos a ser utilizados en la agregación, entre ellos se encuentran el valor máximo (max), la suma algebraica (probor) y la suma (sum), siendo este último el más utilizado, al ser una unión de los conjuntos de salida de las reglas evaluadas, seleccionando los máximos valores para cada conjunto difuso resultante.

Por último, el desfusificador mapea resultado de la agregación a números. Varios métodos son usados para la desfusificación, entre los cuales se encuentran el Centroide, menor valor

de los máximos, media de los máximos, etc.; siendo el más utilizado el método Centroide, el cual calcula y retorna el centroide del conjunto difuso agregado.

La siguiente fórmula es la utilizada para calcular el centroide (Scheneider, Kandel, Langholz y Chew, 1996):

$$y_C = \frac{\sum_{x \in X} x\,\mu_A(x)}{\sum_{x \in X} \mu_A(x)} \tag{1.41}$$

El valor del centroide es entonces el valor numérico de salida del sistema difuso. A continuación se ilustra con un ejemplo gráfico (Figura 1.9):

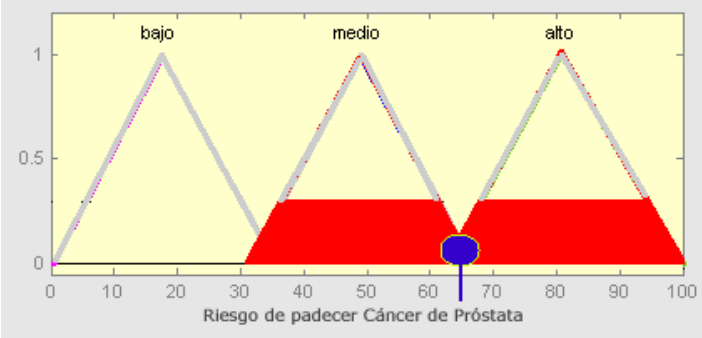



*Figura 1.9. Valor resultante del método de desfusificación "Centroide"*

A continuación se mostrará un ejemplo que ilustra paso a paso todo el proceso de desarrollo previamente mencionado.

Se desea crear un sistema basado en reglas difusas que permita determinar el riesgo de padecer cáncer de próstata según la edad del paciente y su nivel de antígeno específico de la próstata (PSA). Se inicia el proceso con el siguiente paso:

1. Definición de la base de conocimiento y método de implicación:

A partir de datos dados o información extraída de expertos, se crean las reglas difusas que relacionan las variables de entrada con las variables de salida. El antecedente de cada regla cubre cierto rango de valores de entrada, a los que se asocia la misma salida (el consecuente de la regla). En caso de que el antecedente de cierta regla esté compuesto de más de una sentencia, se ejecuta el operador que está asociado a dichas sentencias para

calcular el valor que toma este antecedente. El método de implicación a utilizar es la inferencia max-min, o inferencia de Mamdani.

La Tabla 1.4 ilustra las reglas a utilizar para determinar el riesgo de padecer cáncer de próstata:

*Tabla 1.4. Reglas para determinar el riesgo de padecer cáncer de próstata*

| | Riesgo según la edad | | | | | |
|---|---|---|---|---|---|---|
| Nivel de PSA | | Muy Bajo | Bajo | Medio | Alto | Muy Alto |
| | Muy Bajo | Bajo | Bajo | Bajo | Bajo | Bajo |
| | Bajo | Bajo | Bajo | Bajo | Bajo | Medio |
| | Medio | Bajo | Bajo | Medio | Alto | Alto |
| | Alto | Bajo | Medio | Alto | Alto | Alto |
| | Muy Alto | Medio | Alto | Alto | Alto | Alto |

Tanto el antecedente como el consecuente no son valores numéricos exactos, sino conceptos vagos modelados mediante conjuntos difusos. En los siguientes gráficos (Figuras 1.10, 1.11 y 1.12) se representan los conjuntos difusos asociados a las variables de entrada y de salida del ejemplo.

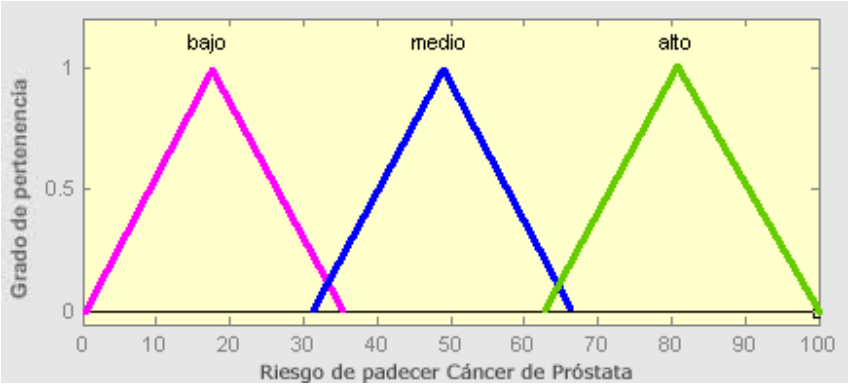



*Figura 1.10. Conjuntos difusos asociados a la variable de salida "Riesgo de padecer cáncer de próstata"*


*Figura 1.11. Resultado del proceso de fusificación de la variable de entrada "Riesgo según edad"*

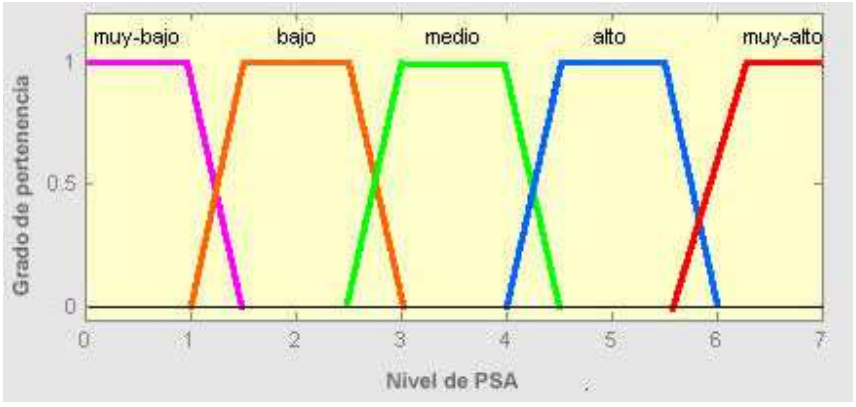



*Figura 1.12. Resultado del proceso de fusificación de la variable de entrada "Nivel de PSA"*

2. Fusificación de las variables de entrada:

A continuación se fusifican las variables de entrada, determinando el grado en el que cada una de ellas pertenece al conjunto difuso apropiado a través de las funciones de pertenencia.

Si como entrada el valor de PSA es igual a 4.3, se puede concluir que pertenece en un grado igual a 0.4 al conjunto difuso "medio" y de 0.6 al conjunto difuso "alto". Por lo tanto se fusifica el valor numérico de entrada (4.3) a 0.4 ó 0.6, los cuales representan los grados de pertenencia al conjunto difuso correspondiente, como se puede observar en el siguiente gráfico (Figura 1.13):

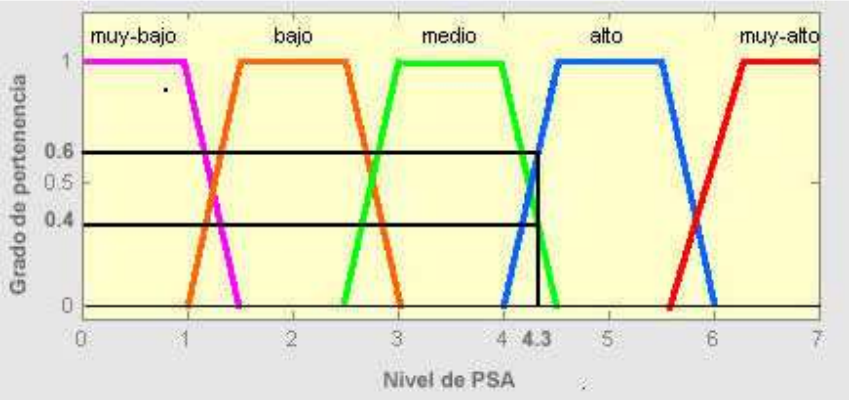



*Figura 1.13. Grados de pertenencia asociados a los conjuntos difusos que contienen el valor x=4.3*

Para el valor asociado al riesgo según edad, se tiene que es igual a 42, se puede concluir que pertenece en un grado igual a 0.7 para el conjunto difuso “bajo” y 0.25 al conjunto difuso “medio”, como se puede observar en el siguiente gráfico (Figura 1.14):


*Figura 1.14. Grados de pertenencia asociados a los conjuntos difusos que contienen el valor x=42*

3. Evaluación de las reglas difusas:

A continuación se evalúan las reglas correspondientes, trabajando sobre la base de conocimientos y utilizando el método de implicación seleccionado previamente. Para esto se comparan las variables de entrada con cada uno de los antecedentes de las reglas, para activar aquellas que coincidan, como se puede observar en la Tabla 1.5:

*Tabla 1.5. Evaluación de las sentencias aplicando el operador Y*

| | Riesgo según la edad | | | | | |
|---|---|---|---|---|---|---|
| Nivel de PSA | | Muy Bajo | 0.7 | 0.25 | Alto | Muy Alto |
| | Muy Bajo | Bajo | Bajo | Bajo | Bajo | Bajo |
| | Bajo | Bajo | Bajo | Bajo | Bajo | Medio |
| | 0.4 | Bajo | 0.4 | 0.25 | Alto | Alto |
| | 0.6 | Bajo | 0.6 | 0.25 | Alto | Alto |
| | Muy Alto | Medio | Alto | Alto | Alto | Alto |

4. <u>Proceso de inferencia:</u>

Siendo utilizada la implicación de Mamdani, cada regla arroja como resultado el menor valor de entre los presentes en sus antecedentes.

Por ejemplo, dados los valores de entrada EDAD igual a 42 y PSA igual a 4.3, se activarían las siguientes reglas:

- R1: SI PSA ES medio Y EDAD ES baja ENTONCES RIESGO ES bajo
- R2: SI PSA ES medio Y EDAD ES media ENTONCES RIESGO ES medio
- R3: SI PSA ES alto Y EDAD ES baja ENTONCES RIESGO ES medio

R4: SI PSA ES alto Y EDAD ES media ENTONCES RIESGO ES alto

Se puede observar gráficamente el proceso de implicación en los siguientes gráficos (Figuras 1.15, 1.16, 1.17, 1.18):


*Figura 1.15. Implicación para la regla R1.*

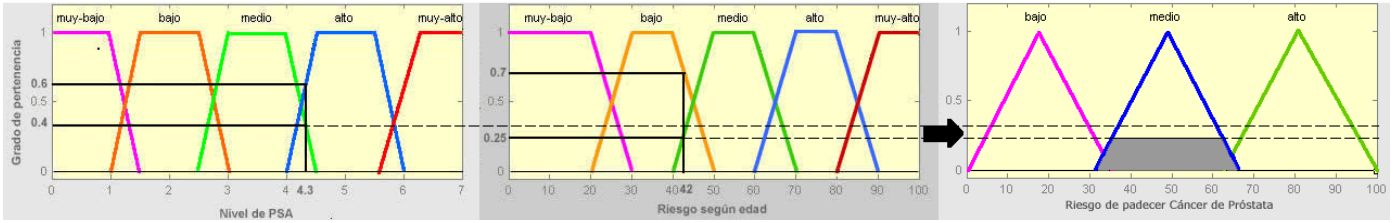



*Figura 1.16. Implicación para la regla R2.*

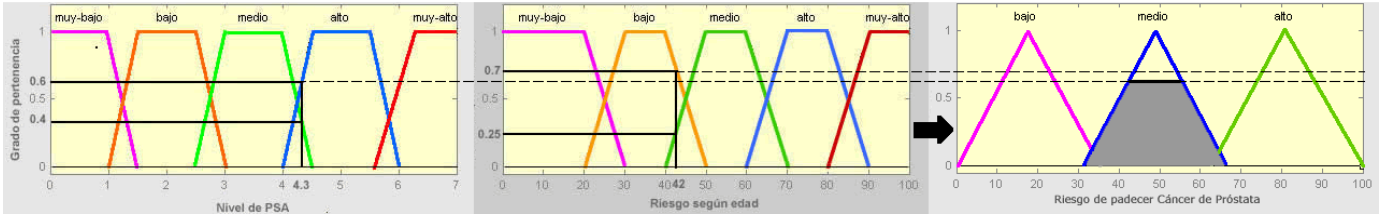



*Figura 1.17. Implicación para la regla R3.*

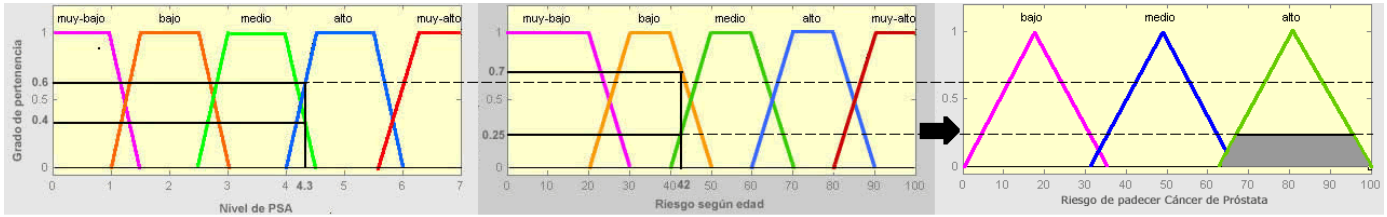



*Figura 1.18. Implicación para la regla R4.*

5. Proceso de agregación:

Como próximo paso se debe ejecutar el proceso de agregación sobre los conjuntos difusos de salida de cada regla, utilizando la función SUM. A continuación se observará gráficamente el resultado (Figura 1.19):

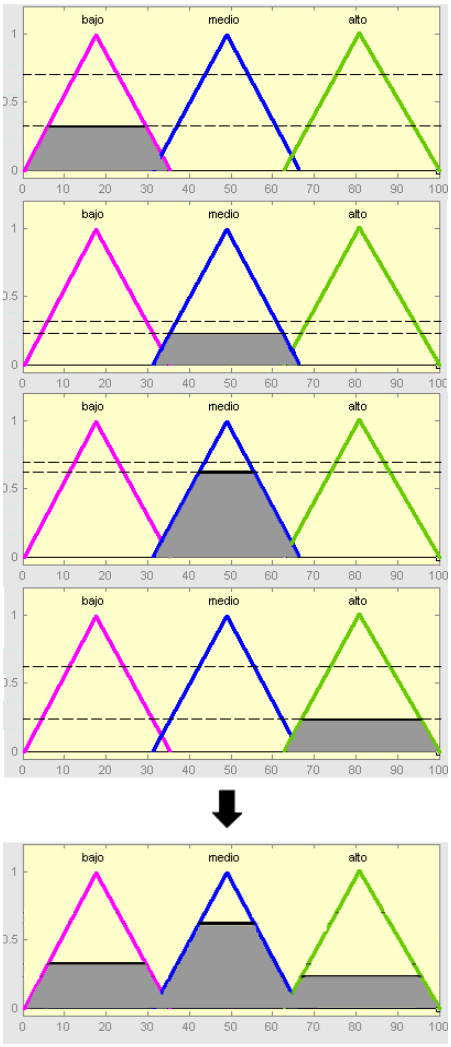



*Figura 1.19. Proceso de agregación sobre los conjuntos difusos de salida.*

6. Desfusificación de los valores de salida:

Finalmente, se deben desfusificar los valores de salida. En este caso es un solo valor que indica el riesgo de padecer cáncer de próstata para un paciente de 42 años de edad y con un valor de PSA de 4.3

Para esto se usará el método del centroide, el cual da como resultado el valor de 44.6, es decir, el riesgo del paciente de padecer cáncer de próstata es de 44.6%.

## 1. 2 Herramientas para la resolución de problemas con lógica difusa

Debido a la creciente popularidad del uso de la lógica difusa para diversas aplicaciones, existe un alto número de herramientas orientadas al desarrollo de soluciones basadas en lógica difusa.

En este capítulo se hará una revisión de algunas herramientas seleccionadas, con el fin de conocer distintas características tales como su interfaz, facilidad de uso, plataforma tecnológica, licencia y funcionalidades.

### Matlab Fuzzy Logic Toolbox – *The MathWorks* (MATLAB, 2009):

Este paquete se utiliza sobre el reconocido software para aplicaciones matemáticas MATLAB. Esto le permite al usuario modificar el código fuente del paquete, agregar sus propias funciones de pertenencia, técnicas de desfusificación, etc.

Posee una sencilla e intuitiva interfaz gráfica que permite realizar todas las operaciones asociadas a la creación de sistemas difusos con facilidad. Como referencia se puede observar la pantalla de inicio a continuación (Figura 1.20).

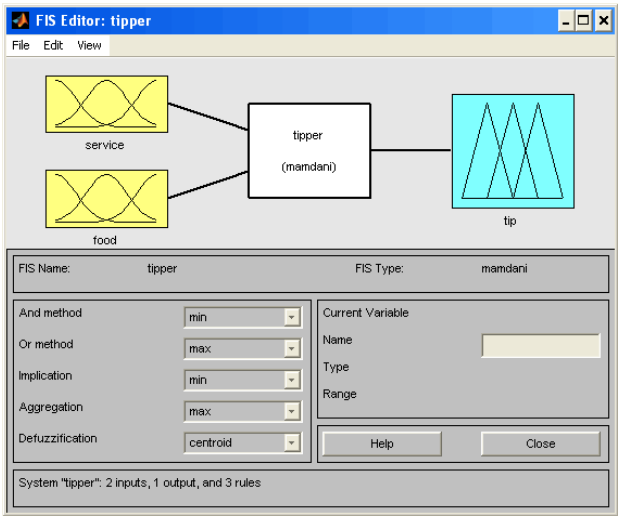



*Figura 1.20. Menú principal de la herramienta de Lógica Difusa de MATLAB (MATLAB, 2009)*

Para poder utilizar este paquete es necesario tener instalado el software MATLAB de The MathWorks, cuya última versión es la 7.8, disponible en versiones para Windows, Linux y

MAC. Cuenta con licencias para estudiantes, profesores y uso comercial. La licencia de estudiantes referente al uso de MATLAB más el paquete Fuzzy Logic tiene un costo aproximado a los 160 dólares americanos.

Entre sus principales funcionalidades se encuentran:

- Utilización de la inferencia de tipo Mamdani y de tipo Sugeno.
- Generación de código C para ser embebido en otras aplicaciones, o motores de inferencia difusas ejecutables.
- Permite la utilización de métodos creados por el usuario o seleccionar diversos métodos ya creados para:
    - los operadores Y (min, prod), O (max, probor)
    - el proceso de implicación (min, prod) y de agregación (max, sum, probor)
    - el proceso de desfusificación (centroide, bisector, mom, lom, som)
- Cuenta con un editor para las funciones de pertenencia (Figura 1.21), que permite colocar el rango, el tipo de la función de pertenencia, su nombre y los parámetros para cada una de ellas, asi como también ver la representación gráfica de los conjuntos difusos.

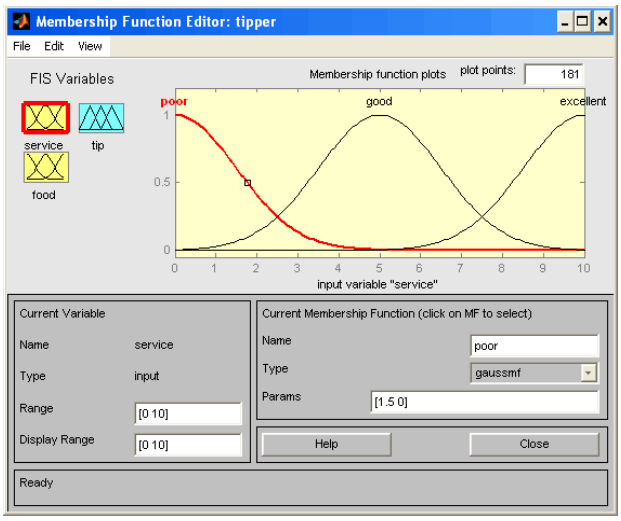



*Figura 1.21. Editor de las funciones de pertenencia (MATLAB, 2009)*

- Cuenta con un editor de reglas (Figura 1.22), donde se pueden agregar, editar y eliminar las reglas del sistema, así como también visualizar las conjuntos difusos

asociados a los antecedentes de cada regla, con sus conjuntos difusos de salidas (asociados a los consecuentes de cada regla) (Figura 1.23).

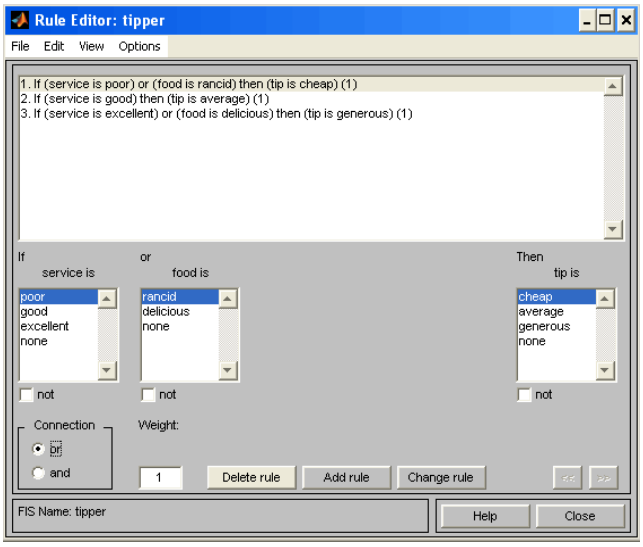



*Figura 1.22. Editor de reglas (MATLAB, 2009).*

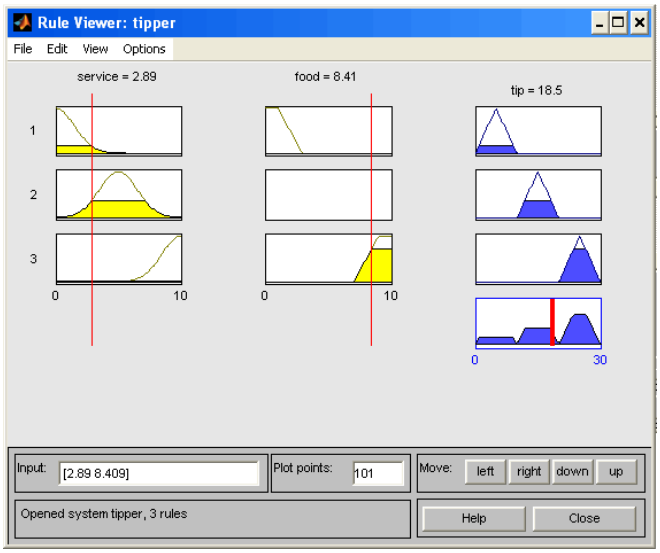



*Figura 1.23. Visor de reglas (MATLAB, 2009).*

## FuzzyCLIPS – NRC Institute for Information Technology, Canada (FuzzyCLIPS, 2009):

FuzzyCLIPS es una versión extendida del lenguaje basado en reglas CLIPS, diseñado para desarrollar sistemas basados en reglas difusas; fue creado por el Grupo de Razonamiento Integrado del Instituto de Tecnología de la Información del Consejo de Investigación Nacional de Canadá.

Su última versión 6.10c, está disponible para las plataformas Windows, Unix, Vax y Solaris. Puede ser descargado gratuitamente junto con la licencia de uso no comercial.

Su interfaz gráfica es bastante limitada al ser una aplicación que se ejecuta en modo consola, su sintaxis es compleja, sin embargo es una herramienta potente. A continuación observa la interfaz gráfica (Figura 1.24), la cual está totalmente basada en caracteres alfanuméricos:

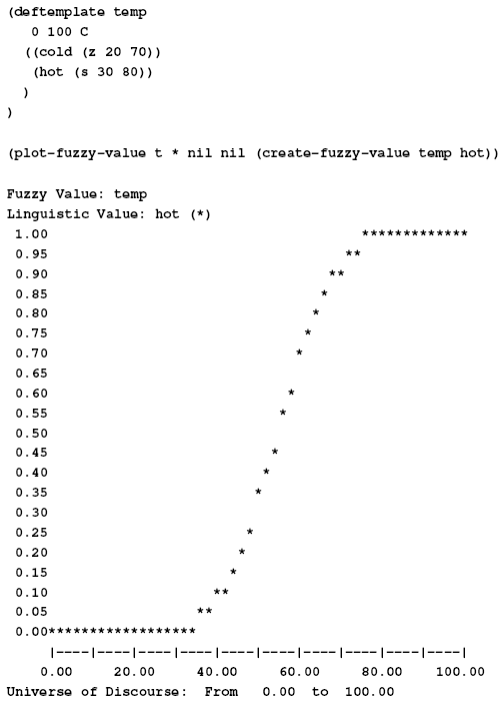



*Figura 1.24. Interfaz FuzzyCLIPS (FuzzyCLIPS, 2009).*

Entre sus principales funcionalidades se encuentran:

- Cuenta con la propiedad llamada “Factor de certeza” el cual es asociado a cada sentencia de una regla o a la regla. Este factor indica con que certeza será verdadera la sentencia o la regla.
- Se pueden definir reglas de razonamiento exacto, reglas difusas o una combinación de ellas.
- Utiliza como método de implicación al método de Mamdani (max-min) y al de Larsen (max-prod).
- Permite el uso de múltiples consecuentes en una sentencia, así como también el uso de múltiples antecedes.

- Es posible configurar una variable llamada “umbral del factor de certeza” para que ninguna regla sea activada a menos que su factor de certeza sea mayor o igual al valor del umbral. Esto permite prevenir que sean activadas reglas con certeza muy baja.
- El proceso de desfusificación utiliza los métodos centroide y mean of maxima (mom).
- Cuenta con funciones de pertenencia predeterminadas y modificadores linguísticos predeterminados tales como not, very, more-or-less, extremely, slightly, etc. Permite que el usuario defina sus propias funciones de pertenencia y modificadores lingüísticos.

## XFuzzy – Instituto de Microelectrónica de Sevilla (Xfuzzy, 2001):

Xfuzzy es un entorno de diseño que permite la construcción de sistemas difusos, está conformado por distintas herramientas que cubren las diferentes etapas del proceso de diseño de un sistema basado en reglas difusas. Xfuzzy está basado en el lenguaje XFL (XFL, 2001), el cual es un lenguaje para la definición de este tipo de sistemas. Este lenguaje fue diseñado en la Universidad de Sevilla, España.

Su última versión 2.1 está disponible para Windows y Unix, bajo licencia GNU.

Posee una sencilla e intuitiva interfaz gráfica que permite realizar todas las operaciones asociadas a la creación de sistemas difusos con facilidad, como se puede observar en la siguiente imagen (Figura 1.25).

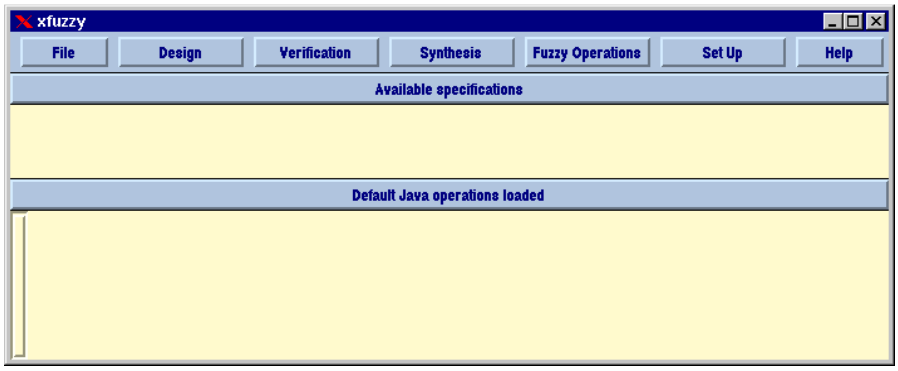



*Figura 1.25. Menú principal de Xfuzzy (Xfuzzy, 2001).*

Entre sus principales funcionalidades se encuentran:

- La definición de las operaciones difusas están predeterminadas para C, Java, entre otros lenguajes de programación. Permite la modificación de las mismas o la creación de nuevas operaciones (Figura 1.26).

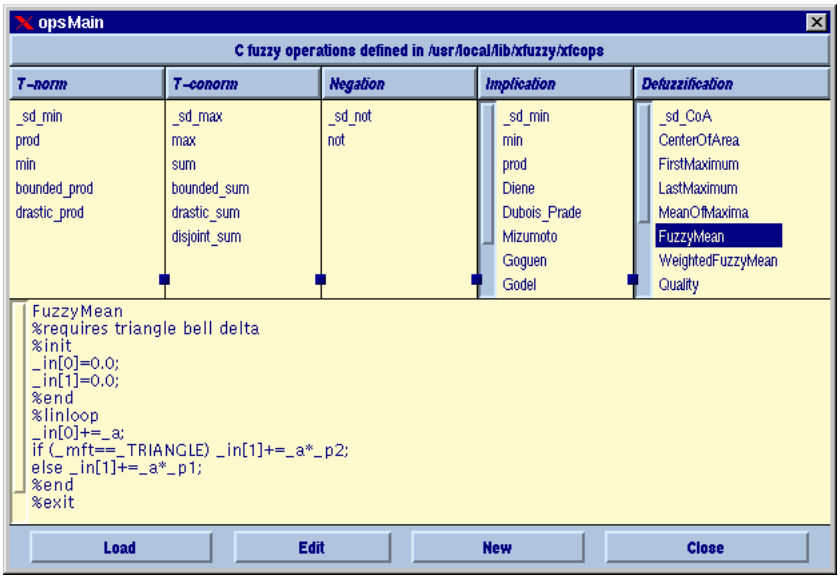



*Figura 1.26. Editor de operaciones (Xfuzzy, 2001).*

- Permite utilizar funciones de pertenencia predeterminadas asi como también modificarlas o crear funciones nuevas (Figura 1.27).

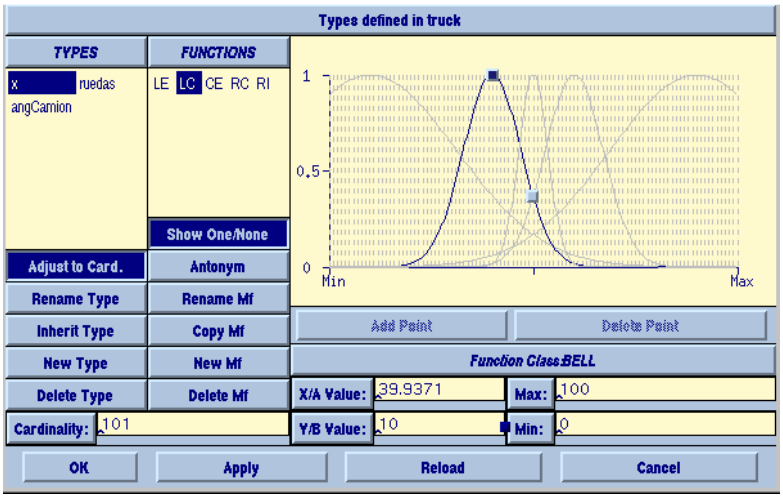



*Figura 1.27. Editor de funciones de pertenencia (Xfuzzy, 2001).*

- Ofrece un módulo de simulación del sistema difuso que cuenta con diferentes visualizadores para observar los resultados (Figura 1.28).

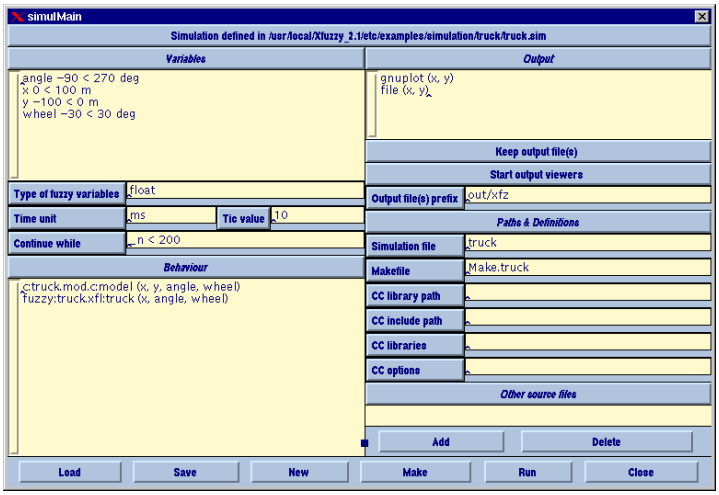



*Figura 1.28. Editor de variables (Xfuzzy, 2001).*

- Entre sus funcionalidades más resaltantes, se encuentra la existencia de un módulo de aprendizaje supervisado (Figura 1.29), el cual permite ajustar la función de error, dando más peso a aquellas variables de salida que se consideren más significativas en la desviación del sistema. También cuenta con un umbral, el cual puede ser configurado por el usuario, que sirve de guía para eliminar reglas cuyo grado de activación no superen dicho umbral. Se puede seleccionar cualquiera de los siguientes algoritmos de aprendizaje: Basado en la regla de Manhattan, Backpropagation con razón de aprendizaje constante, Backpropagation con tamaño de paso constante, R-prop y Quick-prop. También permite al usuario seleccionar las variables y funciones de pertenencia que van a ser ajustados y así probar diferentes estrategias de aprendizaje.

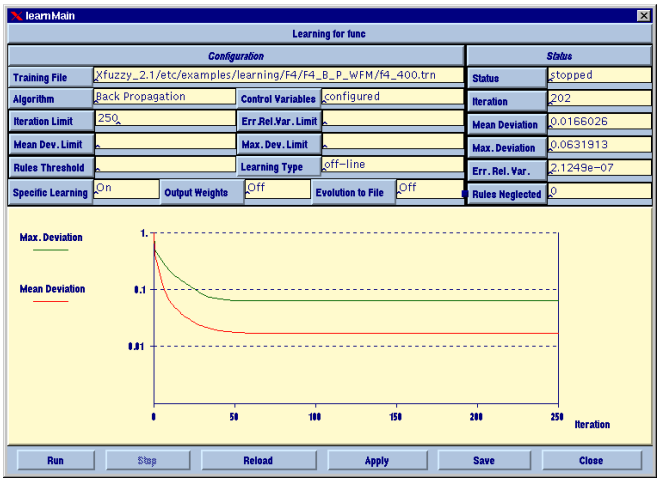



*Figura 1.29. Módulo de aprendizaje (Xfuzzy, 2001).*

- Posee un módulo de síntesis de XFL a otros lenguajes altamente utilizados como lo son C y Java.
- Cuenta con un módulo para la representación gráfica de las superficies asociadas a la salida del sistema (Figura 1.30).

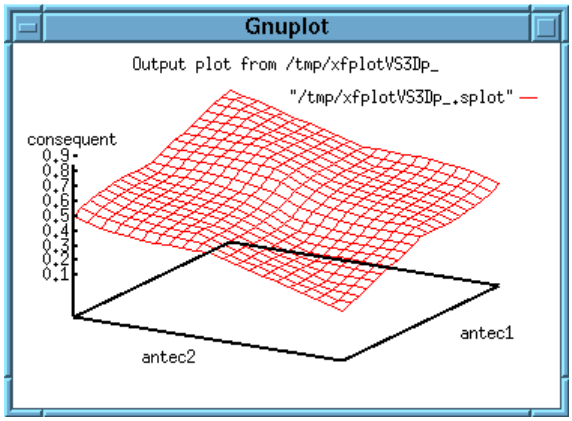



*Figura 1.30. Visor de superficies (Xfuzzy, 2001).*

## fuzzyTECH – *INFORM* (fuzzyTECH, 2009):

fuzzyTECH es una aplicación que permite diseñar sistemas basados en reglas difusas y está basado en el lenguaje FTL. Fue creado por la compañía alemana INFORM.

Existen diversas ediciones, módulos y paquetes, entre los que se encuentran: Professional Edition, MCU Pack for Embedded Control (para diversas marcas y modelos de

microcontroladores), IA Editions for Industrial Automation, NeuroFuzzy Module, HyperInference Module.

Ofrecen la posibilidad de descargar gratuitamente una versión de demostración a través de su sitio web, la cual contiene 12 aplicaciones de ejemplo que pueden ser analizadas y modificadas.

Su última versión 5.72 se ejecuta sólo sobre sistemas operativos Windows y con Internet Explorer con versión igual o superior a la 5.01.

Su intuitiva interfaz gráfica sigue los lineamientos de las interfaces para aplicaciones Windows (Figura 1.31).

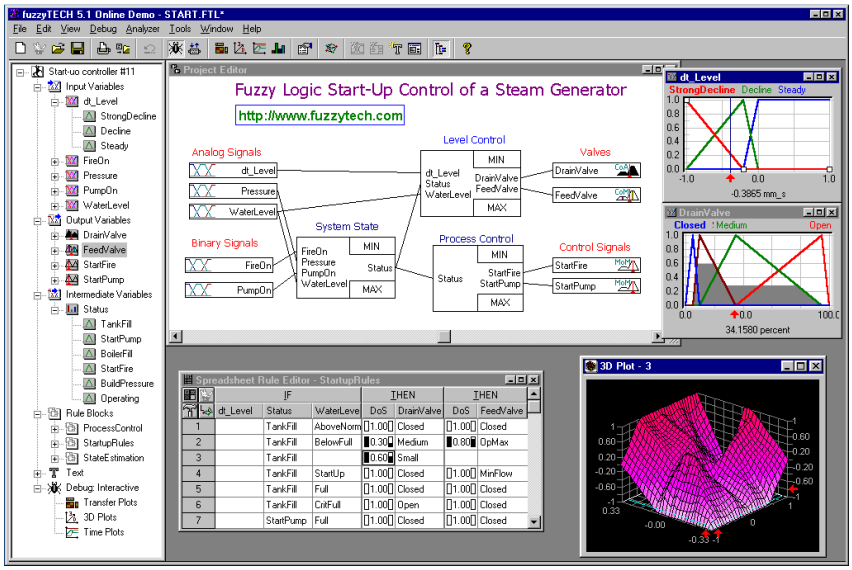



*Figura 1.31. Menú principal de fuzzyTECH (fuzzyTECH, 2009).*

Entre sus principales funcionalidades se encuentran:

- Contiene distintos asistentes para la creación paso a paso de sistemas difusos.
- Posee 3 maneras para editar las reglas: por hoja de cálculo, por matrices o usando el lenguaje FTL (Figura 1.32).

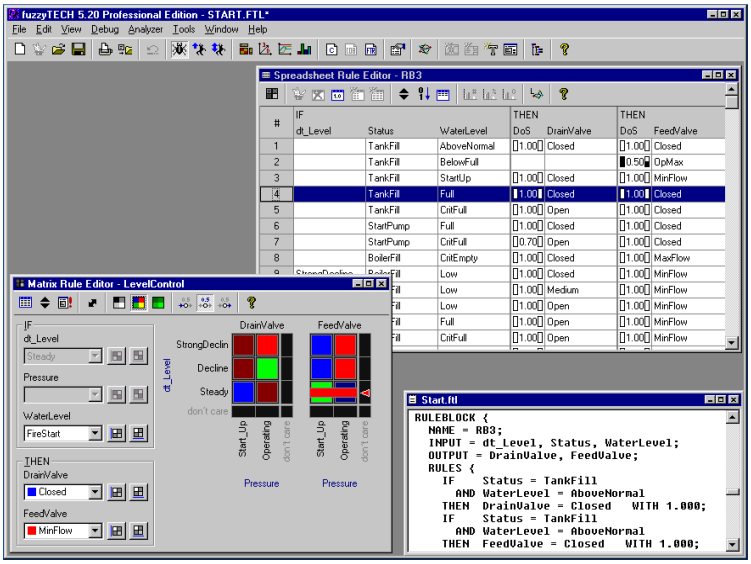



*Figura 1.32. Distintas interfaces para la edición de reglas (fuzzyTECH, 2009).*

- Cuenta con un potente módulo de visualización gráfica y simulación, que permite un completo análisis de los resultados del sistema (Figura 1.33).

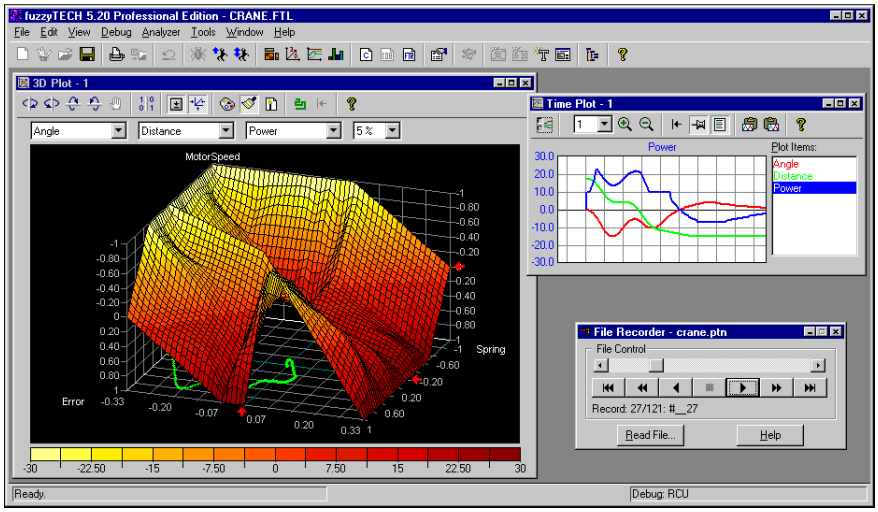



*Figura 1.33. Visor de superficies (fuzzyTECH, 2009).*

- Posee un generador de documentación para el sistema en desarrollo (Figura 1.34), que sigue la norma ISO 9000 y el “IEC 1131-7 standard on fuzzy logic development”.

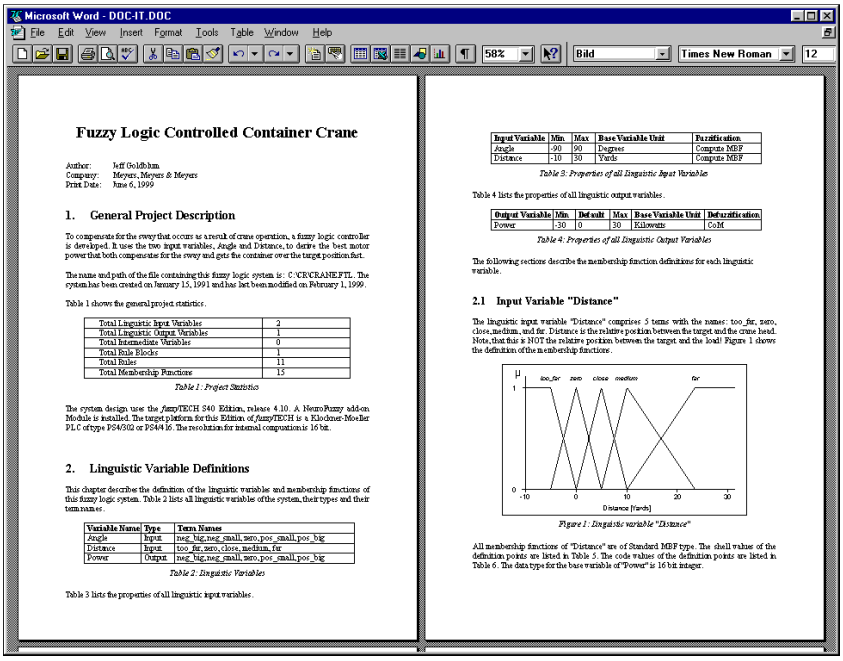



*Figura 1.34. Documentación generada automáticamente (fuzzyTECH, 2009).*

- Puede producir código fuente en lenguaje C, lenguaje ensamblador para distintos microcontroladores.
- Utiliza los métodos de inferencia max-min y max-prod. También utiliza la inferencia FAM (Fuzzy Associative Map).
- Además de utilizar el operador max para el resultado de la agregación, también utiliza el operador BSUM (Bounded-Sum).
- Utiliza tanto el encadenamiento de reglas hacia adelante como el encadenamiento hacia atrás, siendo esto transparente para el usuario ya que fuzzyTECH decide automáticamente cuál es el método de procesamiento adecuado para el sistema en desarrollo.
- Posee operadores predeterminados tales como min-max, avg-max y gamma; sin embargo permite al usuario crear nuevos operadores.

Concluyendo la revisión de las herramientas seleccionadas para este análisis, se puede determinar que aquellas que poseen mayores funcionalidades y mejores interfaces en general no son de libre distribución, por lo que se plantea como objetivo de este trabajo el desarrollo de una herramienta que cuente con las principales funcionalidades y excelente usabilidad, con la ventaja de ser de libre distribución.

# CAPÍTULO II: MARCO APLICATIVO

En este capítulo se presenta el marco aplicativo, el cual está compuesto por el planteamiento del problema, los objetivos, la descripción del desarrollo del prototipo para el desarrollo de aplicaciones basadas en lógica difusa junto con los experimentos realizados y sus resultados.

## 2.1 Planteamiento del problema

La creciente popularidad de la lógica difusa se debe a su buena adaptación a problemas donde existen valores numéricos difíciles de medir, incertidumbre, o excesiva complejidad, así como también a su capacidad de resolver problemas de forma potente y precisa y a la variedad de sus aplicaciones. Actualmente, existen numerosas herramientas para el desarrollo de aplicaciones basadas en lógica difusa, sin embargo, aquellas que cuentan con un mayor número de funcionalidades y mejor usabilidad no son de libre distribución. Por esta razón, surge la idea de contar con una herramienta que reúna las mejores características y funcionalidades de las herramientas existentes, agrupándolas en un ambiente que cuente con una interfaz intuitiva y de fácil uso y de libre distribución.

## 2.2 Objetivos

## Objetivo General

Desarrollar un prototipo de una herramienta que cumpla con las funcionalidades necesarias para el desarrollo de aplicaciones basadas en lógica difusa, bajo un ambiente gráfico que facilite el trabajo del usuario, con una interfaz agradable e intuitiva que permita guiarlo a través de todos los pasos necesarios para desarrollar su aplicación.

## Objetivos Específicos

Los objetivos específicos para el diseño del prototipo de herramienta para el desarrollo de aplicaciones basadas en lógica difusa “fuzzyHAL”, son los siguientes:

- Determinar los requerimientos funcionales de la herramienta.
- Desarrollar la interfaz de usuario.
- Implementar el código fuente que cubre los requerimientos y funcionalidades de la herramienta.

- Realizar las pruebas necesarias para verificar el correcto funcionamiento de las funcionalidades de la herramienta.

## 2.3 Propuesta de solución

Se decide desarrollar un prototipo de herramienta utilizando la metodología de programación orientada a objetos. Se seleccionó el lenguaje de programación orientado a objetos JAVA, en su más reciente versión estándar: JAVA SE 6, la cual puede ser descargada desde el sitio web http://java.sun.com.

Luego de hacer una revisión a las más importantes herramientas para el desarrollo de sistemas basados en reglas difusas existentes en el mercado, se llegó a la conclusión que la herramienta a desarrollar debe cumplir con los siguientes requerimientos:

- Una interfaz gráfica que permita la facilidad de uso de la herramienta, permitiendo que la aplicación cuente con una óptima usabilidad.
- Contar con las funcionalidades necesarias para construir un sistema basado en reglas difusas.
- La herramienta debe ser de libre distribución.

Para lograr los objetivos propuestos para la interfaz gráfica de la herramienta, la misma está basada en los lineamientos planteados por el Lic. Gustavo Torres, para su Trabajo Especial de Grado llamado "*Integración y Mejoramiento Continuo a la Herramienta de Redes Neuronales del Laboratorio de Inteligencia Artificial*" (Torres, 2006). A dicha interfaz se le aplicaron evaluaciones heurísticas y pruebas de aceptación del usuario, bajo la supervisión del Grupo de Interacción Humano-Computador del Centro de Ingeniería de Software y Sistemas de la Escuela de Computación, Facultad de Ciencias de la Universidad Central de Venezuela.

En la Figura 2.1 se muestra la plantilla básica de la interfaz. Está compuesta principalmente por 3 áreas que se describen a continuación:

- Menú principal: se encuentran los casos de uso principales, los cuales se corresponden con los pasos a seguir para desarrollar una aplicación basada en Redes Neuronales.
- Progreso de trabajo: en esta área se muestran los pasos que ya han sido realizados en el desarrollo del sistema.

- Área de trabajo: es el área donde se maneja la información asociada con cada caso de uso principal. También se muestran los mapas conceptuales asociados a cada tarea.

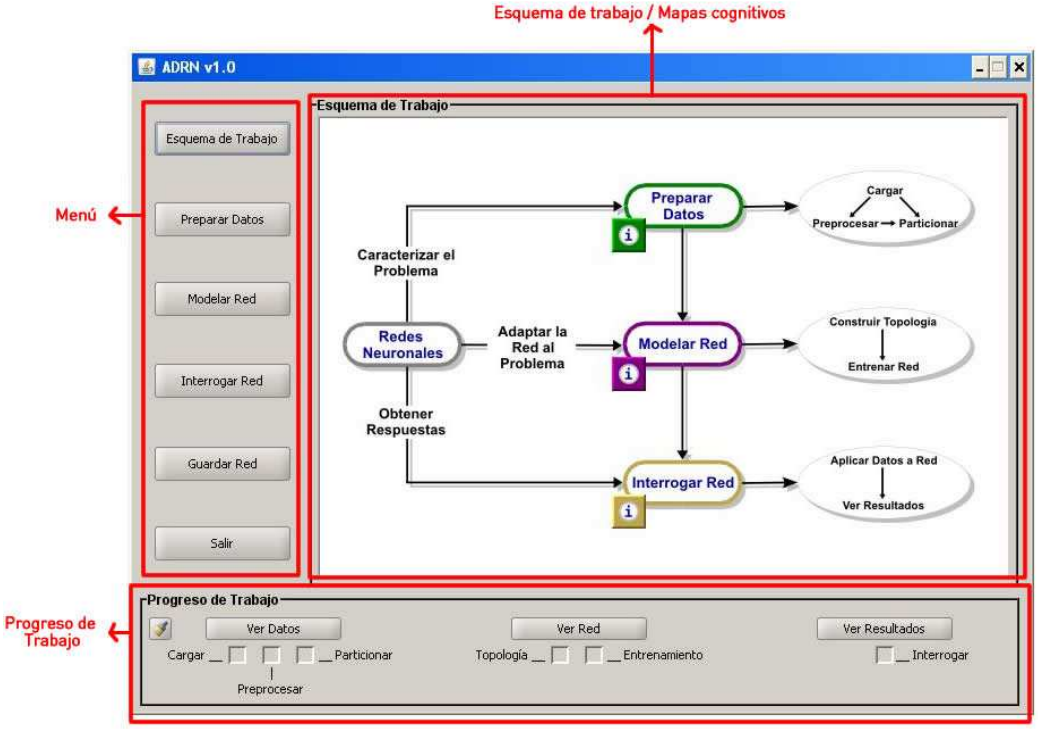



*Figura 2.1. Captura de pantalla de la plantilla básica*

Se analizaron los lineamientos planteados en dicho trabajo, de manera de poder utilizarlos en el prototipo de herramienta a ser desarrollado, garantizando de esta manera que la interfaz del prototipo de herramienta cuente con la usabilidad deseada.

## 2.4 Desarrollo del prototipo para el desarrollo de aplicaciones basadas en lógica difusa

Luego de la revisión de las funcionalidades presentes en las herramientas existentes en el mercado, se describen a continuación los métodos a ser utilizados en la definición de los conjuntos difusos y en el proceso de inferencia:

- Funciones de pertenencia:
  - Triangular
  - Trapezoidal

    - Gaussiana simple
    - Gaussiana doble
    - Campana
    - Tipo Pi
    - Tipo S
    - Tipo Z
    - Sigmoidal
    - Diferencia de Sigmoidales
    - Producto de Sigmoidales
    - Funciones de pertenencia creadas por el usuario
- T-normas:
    - Mínimo (min)
    - Producto Algebraico (prod)
- T-conormas:
    - Máximo (max)
    - Suma Algebraica (probor)
- Métodos de Implicación:
    - Mínimo (min), también conocida como “Implicación de Mamdani”
    - Producto Algebraico (prod), también conocida como “Implicación de Larsen”
- Métodos de Agregación:
    - Máximo (max)
    - Suma (sum)
    - Suma Algebraica (probor)
- Métodos de Desfusificación:
    - Centroide
    - Menor valor de los máximos (SOM)
    - Promedio de los máximos (MOM)
    - Mayor valor de los máximos (LOM)

En el desarrollo del prototipo de herramienta se realiza el análisis de requerimientos, a través del modelo de casos de uso y por último el prototipo de la interfaz gráfica.

### 2.3.1 Análisis de requerimientos y modelos de casos de uso

En la Tabla 2.1 se presenta una breve descripción de los requerimientos levantados.

*Tabla 2.1. Requerimientos funcionales y no funcionales del sistema*

| **Requerimientos funcionales** |
|---|
| - La aplicación permitirá la creación de las variables lingüísticas de entrada y salida.<br>- Permitirá la selección de las funciones de pertenencia asociadas a cada conjunto difuso.<br>- Debe contar con la creación, edición y eliminación de funciones de pertenencia definidas por el usuario.<br>- Debe contar con la creación, edición y eliminación de las reglas difusas.<br>- Permitirá la selección del método de implicación, agregación, desfusificación y los operadores a utilizar.<br>- Permitirá la visualización de resultados del proceso de inferencia.<br>- Se debe contar con la funcionalidad para el almacenamiento del sistema creado, así como también de los resultados generados y que adicionalmente estos datos puedan ser impresos. |
| **Requerimientos no funcionales** |
| - La aplicación tendrá una interfaz intuitiva, con un alto grado de usabilidad.<br>- Contará con la capacidad de ejecutarse sobre cualquier plataforma de software.<br>- Será un software de libre distribución. |

Los casos de usos principales se corresponden con los pasos que se deben seguir para desarrollar un sistema basado en reglas difusas y los necesarios para manipular el sistema:

- Abrir Aplicación: se abre la aplicación para iniciar el desarrollo del sistema.
- Definir Variables Lingüísticas: se crean y editan las variables lingüísticas del sistema, junto con sus conjuntos difusos asociados. Se permite también la creación y edición de nuevas funciones de pertenencia.
- Preparar Base de Conocimientos: se crean y editan las reglas difusas que conforman la base de conocimientos del sistema.

- Iniciar Proceso de Inferencia: se asignan los valores a las variables de entrada al sistema. También se seleccionan los métodos asociados al proceso de inferencia y se inicia el mismo. Al finalizar este proceso se observan los resultados obtenidos.
- Guardar Sistema: se almacenan los datos asociados al sistema desarrollado.
- Cargar Sistema: se cargan los datos de un sistema previamente almacenado.
- Imprimir información del Sistema: se genera un documento con todos los datos del sistema desarrollado, y se solicita al usuario la autorización para imprimirlo.
- Cerrar Aplicación: se cierra la aplicación.

A continuación se presenta el modelo de casos de uso asociados al desarrollo del prototipo.

El usuario de la aplicación puede realizar acciones tales como abrir la aplicación, definir las variables lingüísticas tanto de entrada como de salida, preparar la base de conocimientos, iniciar el proceso de inferencia, visualizar y almacenar los resultados y por último cerrar la aplicación. Los casos de usos asociados y sus descripciones se observan en la Figura 2.2:

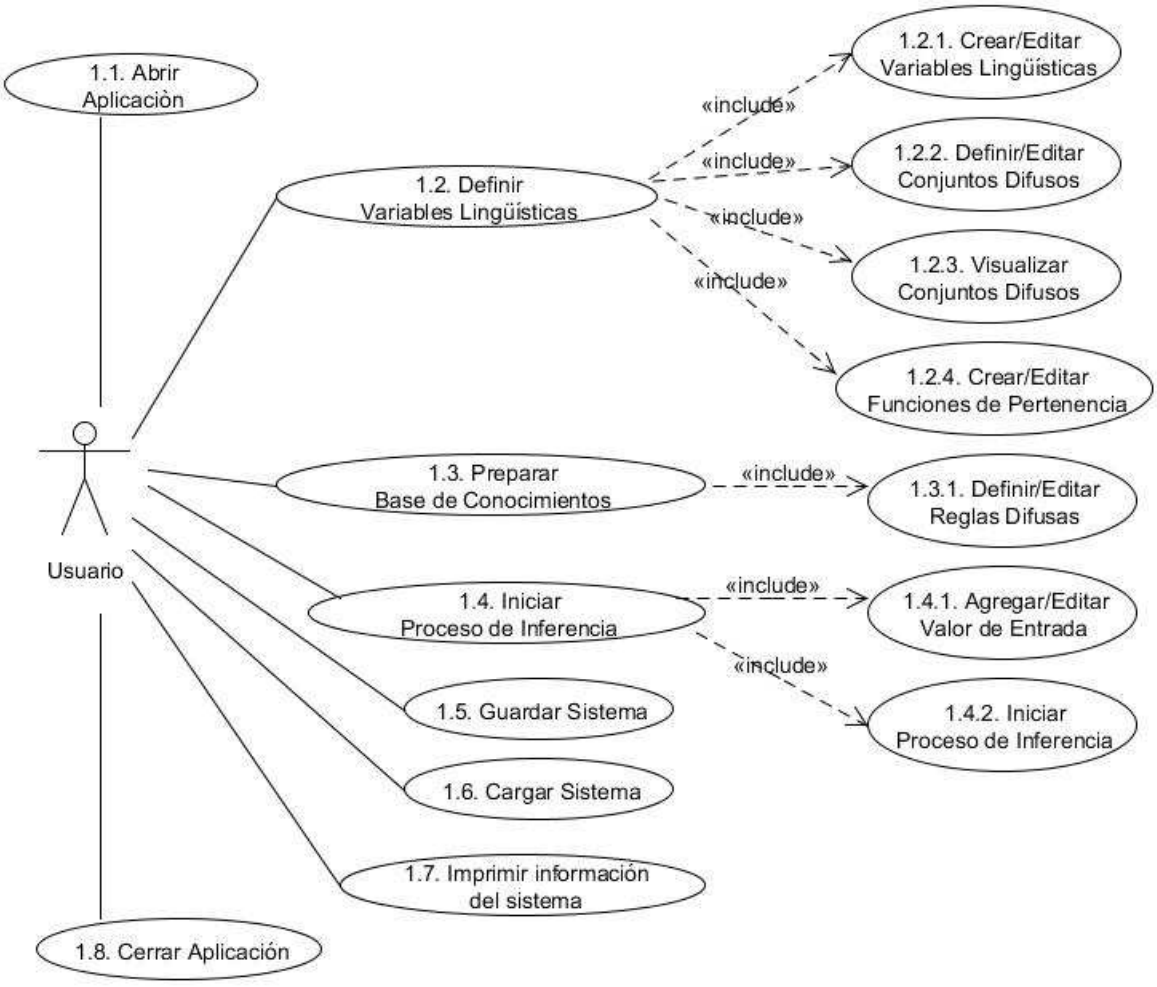



*Figura 2.2. Modelos de casos de uso*

*Tabla 2.2 – Descripción del Caso de Uso "Abrir Aplicación"*

| **ID** | **1.1** |
|---|---|
| **Nombre** | **Abrir aplicación** |
| **Descripción** | El usuario inicia la aplicación |
| **Precondición** | La aplicación aún no ha sido ejecutada |
| **Secuencia** | |
| **Postcondición** | El usuario puede empezar a utilizar la aplicación |

*Tabla 2.3 – Descripción del Caso de Uso "Definir variables lingüísticas"*

| **ID** | **1.2** |
|---|---|
| **Nombre** | **Definir variables lingüísticas** |
| Descripción | El usuario define las variables lingüísticas asociadas al sistema a desarrollar |
| Precondición | Se ha iniciado la aplicación |
| Secuencia | - Se define una variable lingüística<br>- Se crean los conjuntos difusos<br>- Se visualizan los conjuntos difusos definidos<br>- Opcionalmente se puede crear una función de pertenencia |
| Postcondición | El sistema a desarrollar cuenta con las variables lingüísticas y conjuntos difusos asociados |

*Tabla 2.4 – Descripción del Caso de Uso "Preparar base de conocimientos"*

| **ID** | **1.3** |
|---|---|
| **Nombre** | **Preparar base de conocimientos** |
| Descripción | Se crean las reglas difusas que relacionan las variables de entrada con las variables de salida |
| Precondición | Se han definido las variables lingüísticas y conjuntos difusos del sistema a desarrollar |
| Secuencia | - Se define la regla difusa |
| Postcondición | El sistema a desarrollar cuenta con la base de conocimientos |

*Tabla 2.5 – Descripción del Caso de Uso "Iniciar proceso de inferencia"*

| **ID** | **1.4** |
|---|---|
| **Nombre** | **Iniciar proceso de inferencia** |
| Descripción | Se inicia el proceso de inferencia difusa a partir de los datos de entrada dados |
| Precondición | Se han definido las variables lingüísticas, los conjuntos difusos y la base de conocimientos del sistema |
| Secuencia | - El usuario ingresa los valores de entrada al sistema<br>- El usuario selecciona los distintos métodos asociados al proceso de inferencia<br>- El usuario inicia el proceso de inferencia |
| Postcondición | El proceso de inferencia retorna los valores asociados a las variables de salida |

*Tabla 2.6 – Descripción del Caso de Uso "Ingresar valores de entrada"*

| **ID** | **1.4.1** |
|---|---|
| **Nombre** | **Ingresar valores de entrada** |
| Descripción | Se ingresan los valores de entrada al sistema |
| Precondición | El sistema debe estar creado en su totalidad |
| Secuencia | - El usuario ingresa los valores de entrada al sistema |
| Postcondición | El sistema cuenta con valores de entrada para ser evaluados por el proceso de inferencia |

*Tabla 2.7 – Descripción del Caso de Uso "Iniciar proceso de inferencia"*

| **ID** | **1.4.2** |
|---|---|
| **Nombre** | **Iniciar proceso de inferencia** |
| Descripción | Se inicia el proceso de inferencia |
| Precondición | El usuario debe haber ingresado los valores de entrada y haber seleccionado los métodos asociados al proceso de inferencia |
| Secuencia | - El usuario inicia el proceso de inferencia |
| Postcondición | El sistema ejecuta el proceso de inferencia y arroja los resultados |

*Tabla 2.8 – Descripción del Caso de Uso "Guardar Sistema"*

| **ID** | **1.5** |
|---|---|
| **Nombre** | **Guardar Sistema** |
| Descripción | El usuario puede almacenar las características del sistema desarrollado |
| Precondición | Se debe colocar un nombre al sistema a almacenar |
| Secuencia | - El usuario ingresa el nombre del sistema a almacenar |
| Postcondición | El sistema creado es almacenado para uso posterior |

*Tabla 2.9 – Descripción del Caso de Uso "Cargar SIstema"*

| **ID** | **1.6** |
|---|---|
| **Nombre** | **Cargar Sistema** |
| Descripción | El usuario puede cargar un sistema previamente almacenado |
| Precondición | Debe seleccionar el sistema a cargar |
| Secuencia | - Toda la información del sistema cargado se encuentra definida en todas las secciones de la aplicación |
| Postcondición | Se carga en la publicación un sistema seleccionado, el cual ha sido almacenado previamente |

*Tabla 2.10 – Descripción del Caso de Uso "Imprimir información del sistema"*

| **ID** | **1.7** |
|---|---|
| **Nombre** | **Imprimir información del sistema** |
| Descripción | El usuario puede imprimir un documento que contiene toda la información asociada al sistema |
| Precondición | El proceso de inferencia del sistema debe haber arrojado resultados |
| Secuencia | - El usuario selecciona el sistema cuya información desea imprimir |
| Postcondición | El sistema creado, los gráficos generados y sus resultados pueden ser impresos |

*Tabla 2.11 – Descripción del Caso de Uso "Cerrar aplicación"*

| **ID** | **1.8** |
|---|---|
| **Nombre** | **Cerrar aplicación** |
| Descripción | El usuario finaliza la aplicación |
| Precondición | La aplicación debe haber sido iniciada |
| Secuencia | - El usuario finaliza la aplicación |
| Postcondición | La aplicación es finalizada |

El caso de uso “Crear/Editar variables lingüísticas” es descrito gráficamente a continuación, junto a la descripción de cada uno de sus casos de uso asociados (Figura 2.3):

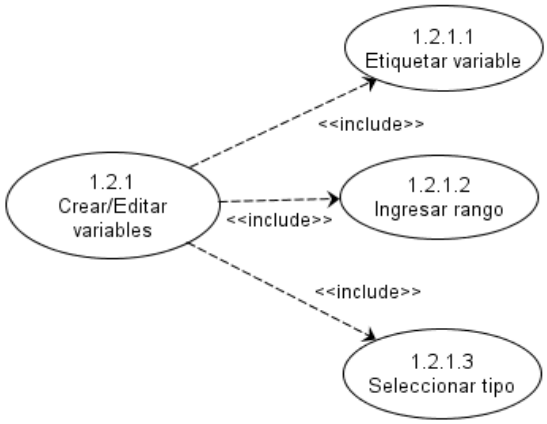



*Figura 2.3. Caso de uso “Crear/Editar variables”*

*Tabla 2.12 – Descripción del Caso de Uso “Crear/Editar variables”*

| **ID** | **1.2.1** |
|---|---|
| **Nombre** | **Crear/Editar variables** |
| Descripción | El usuario crea o edita las variables lingüísticas asociadas al sistema a desarrollar |
| Precondición | Se ha iniciado la aplicación |
| Secuencia | - El usuario crea la variable lingüística<br>- Si el usuario decide editar una variable ya creada, debe seleccionarla para proceder a la edición |
| Postcondición | La variable lingüística es creada o editada |

*Tabla 2.13 – Descripción del Caso de Uso "Etiquetar variables"*

| **ID** | **1.2.1.1** |
|---|---|
| **Nombre** | **Etiquetar variable** |
| Descripción | Se asigna una etiqueta a la variable lingüística creada |
| Precondición | Se debe haber creado una variable lingüística |
| Secuencia | - El usuario asigna una etiqueta a una variable lingüística previamente creada |
| Postcondición | La variable lingüística tiene una etiqueta asociada |

*Tabla 2.14 – Descripción del Caso de Uso "Ingresar rango"*

| **ID** | **1.2.1.2** |
|---|---|
| **Nombre** | **Ingresar rango** |
| Descripción | Se define un rango para la variable lingüística |
| Precondición | Se debe haber creado y etiquetado una variable lingüística |
| Secuencia | - El usuario selecciona la variable lingüística correspondiente<br>- El usuario introduce los valores correspondientes al inicio y final del rango |
| Postcondición | La variable lingüística posee un rango asociado |

*Tabla 2.15 – Descripción del Caso de Uso "Seleccionar tipo"*

| **ID** | **1.2.1.3** |
|---|---|
| **Nombre** | **Seleccionar tipo** |
| Descripción | Se selecciona el tipo de la variable |
| Precondición | La variable lingüística debe estar creada |
| Secuencia | - El usuario indica si la variable seleccionada es de entrada o de salida |
| Postcondición | La variable lingüística tiene un tipo asociado |

El caso de uso “Definir/Editar conjuntos difusos” es descrito gráficamente a continuación, junto a la descripción de cada uno de sus casos de uso asociados (Figura 2.4):

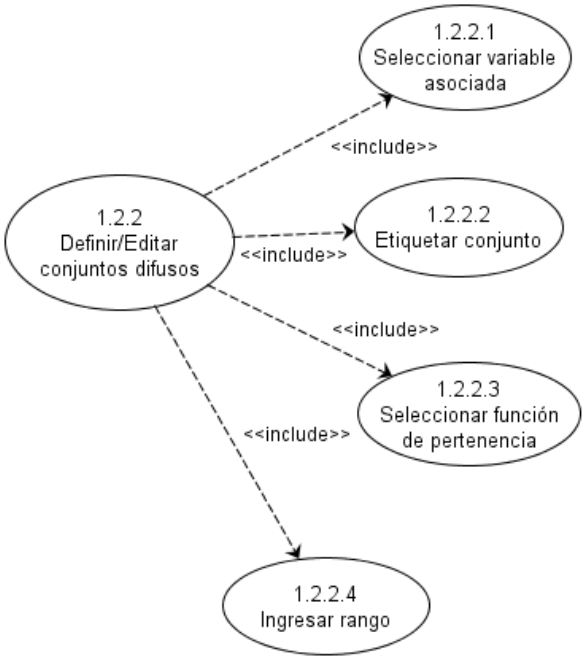



*Figura 2.4. Caso de uso “Definir/Editar conjuntos difusos”*

*Tabla 2.16 – Descripción del Caso de Uso “Definir/Editar conjuntos difusos”*

| **ID** | **1.2.2** |
|---|---|
| **Nombre** | **Definir/Editar conjuntos difusos** |
| Descripción | El usuario define o edita los conjuntos difusos asociados al sistema a desarrollar |
| Precondición | Se han creado las variables lingüísticas |
| Secuencia | - El usuario define el conjunto difuso<br>- Si el usuario decide editar un conjunto difuso ya creado, debe seleccionarlo para proceder a la edición |
| Postcondición | El conjunto difuso es creado o editado |

*Tabla 2.17 – Descripción del Caso de Uso "Seleccionar variable asociada"*

| **ID** | **1.2.2.1** |
|---|---|
| **Nombre** | **Seleccionar variable asociada** |
| Descripción | Se selecciona la variable asociada al conjunto difuso a crear |
| Precondición | Debe existir al menos una variable lingüística creada |
| Secuencia | - El usuario selecciona la variable lingüística |
| Postcondición | El conjunto difuso tiene una variable lingüística asociada |

*Tabla 2.18 – Descripción del Caso de Uso "Etiquetar conjunto"*

| **ID** | **1.2.2.2** |
|---|---|
| **Nombre** | **Etiquetar conjunto** |
| Descripción | Se le asigna un nombre al conjunto difuso |
| Precondición | Se debe haber creado un conjunto difuso |
| Secuencia | - El usuario le asigna una etiqueta al conjunto |
| Postcondición | El conjunto difuso tiene una etiqueta asociada |

*Tabla 2.19 – Descripción del Caso de Uso "Seleccionar función de pertenencia"*

| **ID** | **1.2.2.3** |
|---|---|
| **Nombre** | **Seleccionar función de pertenencia** |
| Descripción | Se selecciona la función de pertenencia asociada al conjunto difuso |
| Precondición | Se debe haber creado un conjunto difuso |
| Secuencia | - El usuario selecciona la función de pertenencia |
| Postcondición | El conjunto difuso tiene una función de pertenencia asociada |

*Tabla 2.20 – Descripción del Caso de Uso "Ingresar rango"*

| **ID** | **1.2.2.4** |
|---|---|
| **Nombre** | **Ingresar rango** |
| Descripción | Se asigna un rango al conjunto difuso |
| Precondición | Se debe haber creado un conjunto difuso |
| Secuencia | - El usuario introduce los valores correspondientes al inicio y final del rango |
| Postcondición | El conjunto difuso tiene un rango asociado |

*Tabla 2.21 – Descripción del Caso de Uso "Visualizar conjuntos difusos"*

| **ID** | **1.2.3** |
|---|---|
| **Nombre** | **Visualizar conjuntos difusos** |
| Descripción | Se asigna un rango al conjunto difuso |
| Precondición | Se debe haber creado un conjunto difuso |
| Secuencia | - El usuario introduce los valores correspondientes al inicio y final del rango |
| Postcondición | El conjunto difuso tiene un rango asociado |

El caso de uso "Crear/Editar Función de Pertenencia" es descrito gráficamente a continuación, junto a la descripción de cada uno de sus casos de uso asociados (Figura 2.5):

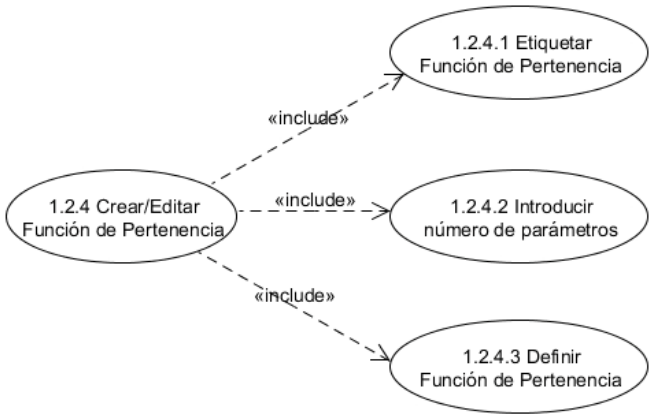



*Figura 2.5. Caso de uso "Crear/Editar Función de Pertenencia"*

*Tabla 2.22 – Descripción del Caso de Uso "Crear/Editar Función de Pertenencia"*

| **ID** | **1.2.4** |
|---|---|
| **Nombre** | **Crear/Editar Función de Pertenencia** |
| Descripción | El usuario crea ó edita funciones de pertenencia |
| Precondición | Se ha iniciado la aplicación |
| Secuencia | - El usuario crea la función de pertenencia<br>- Si el usuario decide editar una función ya creada, debe seleccionarla para proceder a la edición |
| Postcondición | La función de pertenencia es creada o editada |

*Tabla 2.23 – Descripción del Caso de Uso "Etiquetar Función de Pertenencia"*

| **ID** | **1.2.4.1** |
|---|---|
| **Nombre** | **Etiquetar Función de Pertenencia** |
| Descripción | El usuario crea o edita funciones de pertenencia |
| Precondición | Se ha iniciado la aplicación |
| Secuencia | - El usuario crea la función de pertenencia<br>- Si el usuario decide editar una función ya creada, debe seleccionarla para proceder a la edición |
| Postcondición | La función de pertenencia es creada o editada |

*Tabla 2.24 – Descripción del Caso de Uso "Introducir número de parámetros"*

| **ID** | **1.2.4.2** |
|---|---|
| **Nombre** | **Introducir número de parámetros** |
| Descripción | El usuario indica el número de parámetros que tiene la función de pertenencia a definir |
| Precondición | Se ha asignado un nombre a la función de pertenencia |
| Secuencia | - El usuario introduce el número de parámetros |
| Postcondición | La función de pertenencia tiene un número de parámetros asociado |

*Tabla 2.25 – Descripción del Caso de Uso "Definir Función de Pertenencia"*

| **ID** | **1.2.4.3** |
|---|---|
| **Nombre** | **Definir Función de Pertenencia** |
| Descripción | El usuario define la función de pertenencia |
| Precondición | Se ha asignado un nombre y un número de parámetros a la función de pertenencia |
| Secuencia | - El usuario define la función de pertenencia siguiendo la sintaxis del lenguaje de programación Java<br>- La aplicación verifica que la sintaxis esté correcta, en caso contrario el usuario debe corregir la definición de la función |
| Postcondición | La función de pertenencia es definida |

El caso de uso “Definir/Editar reglas difusas” es descrito gráficamente a continuación, junto a la descripción de cada uno de sus casos de uso asociados (Figura 2.6):

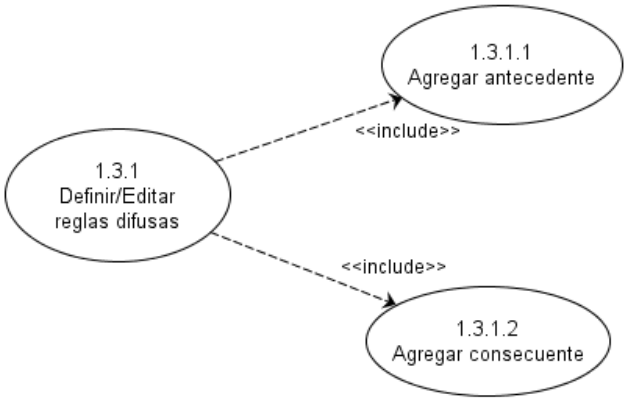



*Figura 2.6. Caso de uso “Definir/Editar reglas difusas”*

*Tabla 2.26 – Descripción del Caso de Uso “Definir/Editar reglas difusas”*

| **ID** | **1.3.1** |
|---|---|
| **Nombre** | **Definir/Editar reglas difusas** |
| Descripción | El usuario define o edita las reglas difusas asociadas al sistema a desarrollar |
| Precondición | Se han creado las variables lingüísticas y los conjuntos difusos |
| Secuencia | - El usuario define el antecedente de la regla<br>- En caso de tener más de un antecedente, se selecciona el operador a utilizar<br>- El usuario define el consecuente de la regla |
| Postcondición | La regla difusa es creada o editada |

*Tabla 2.27 – Descripción del Caso de Uso "Agregar antecedente"*

| **ID** | **1.3.1.1** |
|---|---|
| **Nombre** | **Agregar antecedente** |
| Descripción | Se agrega un antecedente a la regla difusa a crear |
| Precondición | Deben haberse creado los conjuntos difusos asociados al sistema |
| Secuencia | - El usuario selecciona el valor para la variable lingüística correspondiente<br>- El usuario puede agregar más antecedentes, seleccionando previamente el conector lógico a utilizar |
| Postcondición | La regla difusa cuenta con el antecedente asociado |

*Tabla 2.28 – Descripción del Caso de Uso "Agregar consecuente"*

| **ID** | **1.3.1.2** |
|---|---|
| **Nombre** | **Agregar consecuente** |
| Descripción | Se agrega el consecuente con la regla difusa a crear |
| Precondición | Se debe haber creado el antecedente de la regla difusa |
| Secuencia | - El usuario selecciona el valor para la variable lingüística correspondiente |
| Postcondición | La regla difusa cuenta con el consecuente asociado |

## 2.3.2 Arquitectura del sistema

La arquitectura del sistema fue definida en 3 capas, como se puede observar en la Figura 2.7. La primera capa, llamada capa de presentación, se encarga en proveer la interfaz gráfica y de entrada/salida de datos, entre la aplicación y el usuario. La segunda, llamada capa lógica, se encarga de realizar las operaciones asociadas a la información recibida desde la capa de presentación y procesar la información devuelta por la capa de datos. Por último, la tercera capa, llamada capa de datos, es aquella que está compuesta por los archivos que constituyen la base de datos de la aplicación.


*Figura 2.7. Capas que componen la arquitectura del sistema*

**Capa de presentación**

Está compuesta por las clases que se describen en la Tabla 2.29:

*Tabla 2.29 – Descripción de las clases que componen la Capa de Presentación de la herramienta*

| CLASE | DESCRIPCIÓN |
|---|---|
| **InicioUI.java** | Presentación de la aplicación |
| **PrincipalUI.java** | Área principal |
| **VariableConjUI.java** | Se definen las variables lingüísticas, conjuntos difusos y funciones de pertenencia creadas por el usuario |
| **BaseConocimientosUI.java** | Se definen las reglas difusas |
| **ProcesoInferenciaUI.java** | Se define el valor de entrada al sistema y se ejecuta el proceso de inferencia |

| | |
|---|---|
| **VisualizarConjuntos.java** | Se ven las gráficas que representan los conjuntos difusos definidos en el sistema |
| **InferenciaResultados.java** | Se ven los resultados arrojados por el proceso de inferencia |
| **A_CargarSistema.java** | Área donde se carga un sistema previamente almacenado |
| **A_EditarFunPertenencia.java** | Se crean, editan ò eliminan los conjuntos difusos |
| **A_EditarFuncionesPertenencia1.java** | Se crean, editan o eliminan las funciones de pertenencia creadas por el usuario |
| **A_EditarInferencia.java** | Se crea o edita el valor de entrada al proceso de inferencia |
| **A_EditarReglas.java** | Área donde se crean, editan o eliminan las reglas difusas |
| **A_EditarVariables.java** | Se crean, editan o eliminan las variables lingüísticas |
| **A_GuardarSistema.java** | Se guardan todos los datos asociados al sistema creado |
| **A_MapaBaseConocimientos.java** | Muestra el mapa cognitivo asociado a la base de conocimientos |
| **A_MapaProcesoInferencia.java** | Muestra el mapa cognitivo asociado al proceso de inferencia |
| **A_MapaVariables.java** | Muestra el mapa cognitivo asociado a las variables lingüísticas |
| **A_DescFunPertenencia.java** | Muestra la descripción de una función de pertenencia dada |
| **ErroresCompilacion.java** | Muestra los errores de compilación asociados a la función de pertenencia que está siendo creada por el usuario |

**Capa Lógica**

La capa lógica está compuesta por las siguientes 3 clases y sus métodos, descritos en la Tabla 2.30:

*Tabla 2.30 – Descripción de las clases que componen la capa lógica de la herramienta*

| **CLASE: MotorInterfaz** |
|---|
| **MÉTODOS:**<br><br>ActualizarCuadroProgreso: Actualiza el cuadro de progreso principal<br><br>ActualizarTextArea: Actualiza el contenido de un elemento TextArea dado<br><br>ActualizarTextAreaEntrada: Actualiza el contenido del elemento TextArea de los valores de entrada<br><br>ActualizarTextAreaReglas: Actualiza el contenido del elemento TextArea donde aparecen las reglas difusas creadas<br><br>ComboConjunto: Genera un elemento combobox con los conjuntos difusos creados<br><br>ComboFuncionesPertenencia: Genera un elemento combobox con las funciones de pertenencia que existen en el sistema<br><br>ComboSistemas: Genera un elemento combobox con todos los sistemas que han sido almacenados<br><br>ComboVarEntrada: Genera un elemento combobox con todas las variables de entrada del sistema<br><br>ComboVarSalida: Genera un elemento combobox con todas las variables de salida del sistema |
| **CLASE: MotorInferencia** |
| **MÉTODOS:**<br><br>Agmax: Calcula el resultado del proceso de agregación utilizando el operador max<br><br>Agprobor: Calcula el resultado del proceso de agregación utilizando el operador probor<br><br>Agregacion: Realiza el proceso de agregación<br><br>Agsum: Calcula el resultado del proceso de agregación utilizando el operador sum |

CalculoOperadores: Realiza la evaluación de las reglas disparadas, generando los conjuntos resultantes para su posterior agregación

EvaluacionFuncion: Realiza la evaluación de la función de pertenencia dada

Fusificacion: Realiza el proceso de fusificación de la variable de entrada dada

Omax: Calcula el resultado de la ejecución del operador O max

Oprobor: Calcula el resultado de la ejecución del operador O probor

ReglasDisparadas: Realiza la evaluación de las reglas difusas determinando las que han sido disparadas o activadas

Ymin: Calcula el resultado de la ejecución del operador Y min

Yprod: Calcula el resultado de la ejecución del operador Y prod

Centroide: Calcula la salida del proceso de desfusificación del resultado utilizando el método del Centroide

Lom: Calcula la salida del proceso de desfusificación del resultado utilizando el método LOM

Mom: Calcula la salida del proceso de desfusificación del resultado utilizando el método MOM

ordenarParXY: Ordena de manera ascendente los pares (x,y) utilizados para la graficación

Som: Calcula la salida del proceso de desfusificación del resultado utilizando el método SOM

**CLASE: MotorDB**

**MÉTODOS:**

CargarSistema: Carga un sistema previamente almacenado por el usuario

ChequearFuncionCustom: Compila en tiempo de ejecución la función de pertenencia creada por el usuario

ChequearNombreSistema: Chequea si el nombre dado al sistema está siendo utilizado

CrearArchivoFuncion: Se almacena la función de pertenencia creada por el usuario, que fue previamente compilada en busca de errores, en el archivo de funciones de

pertenencia

CrearPDF: Crea un archivo de tipo pdf con toda la información del sistema para su posterior impresión

EditarConjuntoDifuso: Actualiza en la base de datos los nuevos valores asociados a un conjunto difuso que ha sido editado

EditarVariable: Actualiza en la base de datos los nuevos valores asociados a una variable que ha sido editada

EliminarConjuntoDifuso: Elimina de la base de datos un conjunto difuso dado

EliminarRegla: Elimina de la base de datos una regla dada

EliminarVariable: Elimina de la base de datos una variable dada

EscribirArchivo: Escribe datos sobre un archivo .dat

FormatearRegla: Da formato correcto en la base de datos a una regla difusa

FuncionExiste: Verifica si el nombre de una función de pertenencia está siendo utilizado por otra

GuardarConjuntoSalida: Almacena en la base de datos los valores asociados a un conjunto de salida

GuardarEntradaInferencia: Almacena en la base de datos el valor de entrada al proceso de inferencia

GuardarReglaDisparada: Almacena en la base de datos una regla disparada o activada

GuardarSistema: Guarda en la base de datos los valores asociados a un sistema dado

GuardarVariable: Almacena en la base de datos los valores asociados a una variable

Imprimir: Se encarga de ejecutar el proceso de impresión del archivo de extensión “pdf” generado previamente, que contiene la información asociada al sistema creado

LeerArchivo: Lee el contenido de un archivo .dat

LeerArchivoFuncion: Lee el contenido del archivo que contiene las funciones de pertenencia creadas por el usuario

LeerConjunto; Lee de la base de datos los valores asociados a un conjunto difuso

LeerConjuntoSalida: Lee de la base de datos los valores asociados a un conjunto de

salida

LeerConjuntoGraficacion: Lee de la base de datos los valores asociados a un conjunto difuso para su posterior graficación

LeerFuncionConjunto: Lee de la base de datos los valores asociados dados el nombre de la variable y del conjunto difuso

LeerRangoVariable: Lee de la base de datos el rango asociado dado el nombre de la variable

LeerVariables: Lee de la base de datos todas las variables tanto de entrada como de salida

LimpiarCustomTemp: Limpia el contenido del archivo CustomTemp.java, el cual es utilizado para validar la sintaxis de las funciones de pertenencia definidas por el usuario

ProcesarReglas: Recorre las reglas difusas almacenadas y en caso de que alguna posea un operador de negación se generan nuevas reglas derivadas de esta, de manera que no hayan reglas con el operador de negación

RangoFuncionCustom: Dada una función de pertenencia definida por el usuario, se devuelve su rango

conjNegacion: Dada una variable y un conjunto difuso (negado), se devuelven los otros conjuntos difusos asociados.

### Capa de Datos

La capa de datos está constituida por el archivo “db.dat”, el cual almacena todos los datos asociados al sistema, con el formato:

<INICIO_nombreEstructura>

    <dato_1>< dato_2>…< dato_n>

<FIN_nombreEstructura>

Está compuesto por las estructuras descritas en la Tabla 2.31:

*Tabla 2.31 – Descripción de las estructuras que componen la Capa de Datos*

| **ESTRUCTURAS** |
| --- |
| conjuntosResultantes: Almacena los datos asociados a los conjuntos resultantes en el proceso de inferencia.<br><br>datosInferencia: Almacena la información referente a los métodos utilizados en el proceso de inferencia (Y, O, Implicación, Agregación, Desfusificación).<br><br>errores: Almacena los errores de compilación asociados a una función de pertenencia creada por el usuario.<br><br>funcionesCustom: Almacena el nombre y rango de las funciones de pertenencia creadas por el usuario.<br><br>funPertenencia: Almacena los datos asociados a los conjuntos difusos.<br><br>reglaant: Almacena temporalmente los datos de la regla difusa que está siendo creada o editada.<br><br>reglas: Almacena las reglas difusas del sistema.<br><br>reglasDisparadas: Almacena las reglas que fueron disparadas o activadas durante el proceso de inferencia.<br><br>resultado: Almacena los resultados asociados a las variables de salida del sistema.<br><br>valoresInferencia: Almacena los valores de entrada al proceso de inferencia.<br><br>varEntrada: Almacena los valores de entrada al proceso de inferencia.<br><br>varSalida: Almacena los datos asociados a las variables de salida. |

### 2.3.3 Diseño de la interfaz

Como se explicó al inicio de este capítulo, la interfaz gráfica de la herramienta, llamada "fuzzyHAL", está basada en la que fue propuesta por el Lic. Gustavo Torres, para su Trabajo Especial de Grado llamado "*Integración y Mejoramiento Continuo a la Herramienta de Redes Neuronales del Laboratorio de Inteligencia Artificial*" (Torres, 2006).

La interfaz utiliza un estilo llamado "PlasticXP", que pertenece a la librería de código abierto llamada "JGoodies Looks R.1.3.2" (JGoodies, 2005).

La descripción de la interfaz desarrollada se realizará por medio de un ejemplo, donde se mostrarán los pasos necesarios para desarrollar un sistema basado en reglas difusas.

En la Figura 2.8 se muestra la plantilla básica de la interfaz. Sus áreas se encuentran resaltadas con colores para señalizarlas y describirlas a continuación:

- Menú principal: el recuadro rojo delimita el menú donde se encuentran los casos de uso principales, los cuales se corresponden con los pasos a seguir para desarrollar un sistema basado en reglas difusas.
- Progreso de trabajo: el recuadro amarillo delimita esta área, donde se muestran los pasos que ya han sido realizados en el desarrollo del sistema, y adicionalmente se podrán reiniciar los datos asociados al sistema creado.
- Área de trabajo: el recuadro verde delimita el área de trabajo, donde se maneja la información asociada con cada caso de uso principal. También se muestran los mapas conceptuales asociados a cada tarea.

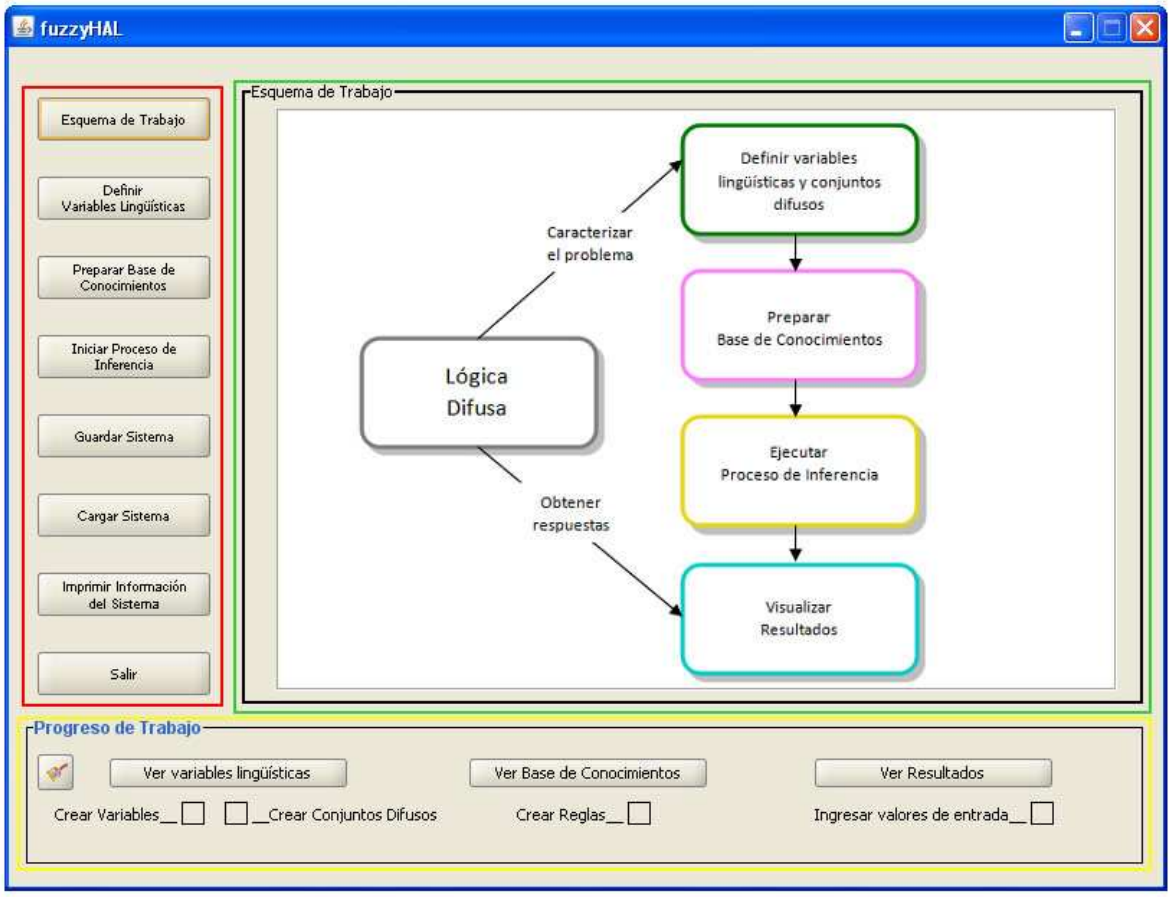



*Figura 2.8. Captura de pantalla de la plantilla básica*

En el menú izquierdo se encuentran los principales casos de uso:

- Esquema de Trabajo: muestra un mapa cognitivo que describe el ciclo de desarrollo de un sistema basado en reglas difusas.
- Definir Variables Lingüísticas: se crean y editan las variables lingüísticas y conjuntos difusos, así como también se pueden crear y editar nuevas funciones de pertenencia.
- Preparar Base de Conocimientos: se crean y editar las reglas difusas.
- Iniciar Proceso de Inferencia: se indica el valor de entrada y los métodos a utilizar en el proceso de inferencia, para posteriormente conocer los resultados.
- Guardar Sistema: se almacenan los datos del sistema creado.
- Cargar Sistema: se cargan en la herramienta los datos de un sistema anteriormente almacenado.
- Imprimir Información del Sistema: permite la impresión de toda la información asociada al sistema.

- Salir: permite la salida de la herramienta.

Se describen a continuación el sistema que se va a crear como ejemplo de utilización de la herramienta, a la vez que se describirá la interfaz. Las características de dicho sistema se describen en la Tabla 2.32:

*Tabla 2.32 – Descripción de las clases que componen la Capa Lógica de la herramienta*

| CARACTERÍSTICA | VALORES |
|---|---|
| **Variables de entrada** | - Velocidad Rpm del motor "velocidad"<br>Rango [2304 2536] |
| **Variables de salida** | - Voltaje "voltaje"<br>Rango [2.32 2.48] |
| **Conjuntos difusos** | - Variable "velocidad":<br>- Nombre del Conjunto "Muy lenta"<br>• Función de pertenencia Triangular<br>• Rango [2304 2362 2420]<br>- Nombre del Conjunto "Normal"<br>• Función de pertenencia Triangular<br>• Rango [2391 2420 2449]<br>- Nombre del Conjunto "Muy rápida"<br>• Función de pertenencia Triangular<br>• Rango [2420 2478 2536]<br>- Variable "voltaje":<br>- Nombre del Conjunto "Bajar"<br>• Función de pertenencia Triangular<br>• Rango [2.32 2.36 2.40]<br>- Nombre del Conjunto "Mantener"<br>• Función de pertenencia Triangular<br>• Rango [2.38 2.40 2.42] |

| | |
|---|---|
| | - Nombre del Conjunto “Subir”<br>  • Función de pertenencia Triangular<br>  • Rango [2.40 2.44 2.48] |
| **Reglas** | - Si la velocidad del motor es muy lenta, entonces subir el voltaje para acelerar<br>- Si la velocidad del motor es normal, entonces mantener el voltaje<br>- Si la velocidad del motor es muy rápida, entonces bajar el voltaje para desacelerar |
| **Métodos asociados al proceso de inferencia** | - Método Y: min<br>- Método O: max<br>- Implicación: min<br>- Agregación: max<br>- Desfusificación: Centroide |

Al iniciar la aplicación, se muestra la siguiente pantalla (Figura 2.9):


*Figura 2.9. Pantalla de inicio de la aplicación*

Dependiendo de la tarea que se vaya a ejecutar el área de trabajo cambia, así como también el área de progreso de trabajo.

El proceso de desarrollo del sistema usado como ejemplo se describe a continuación:

1. Definición de variables lingüísticas y conjuntos difusos

En la Figura 2.10 se observa la pantalla diseñada para la tarea “Definir Variables Lingüísticas”:

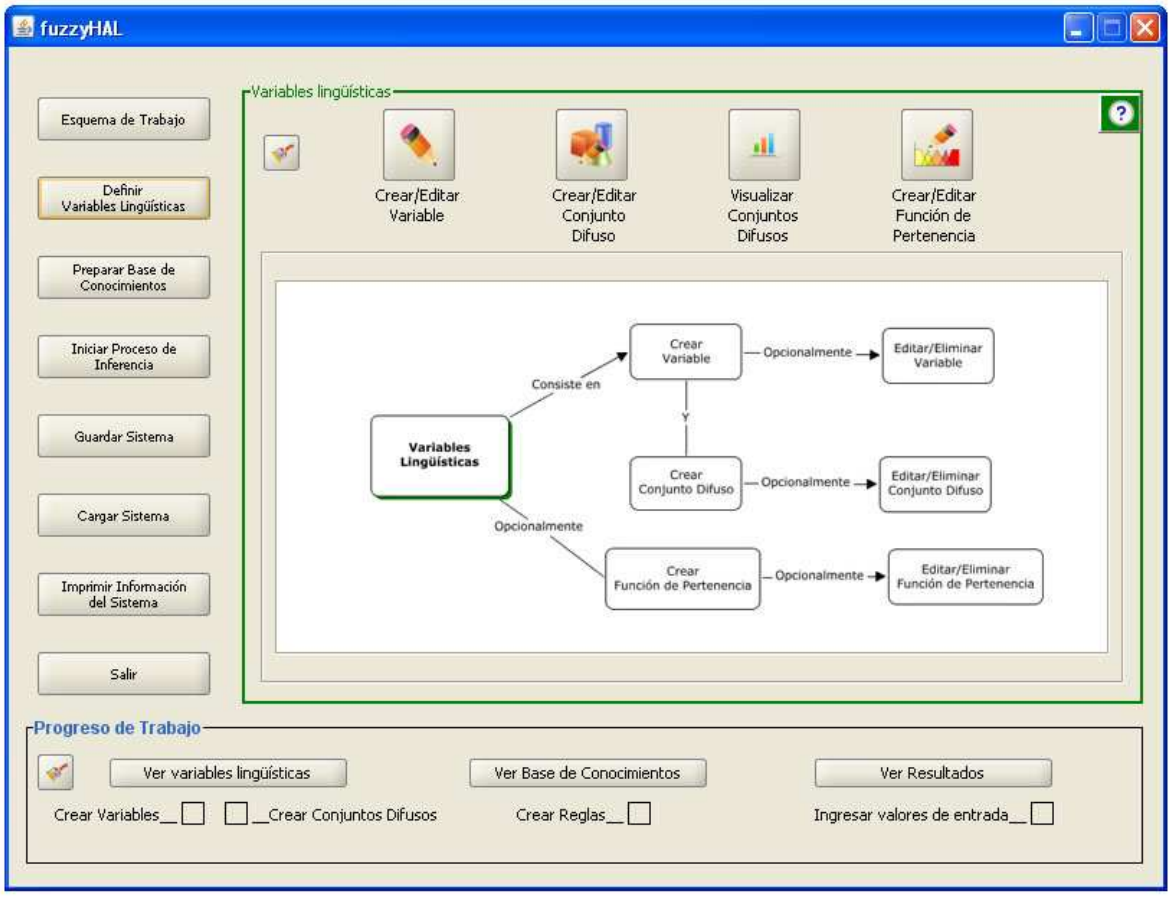



*Figura 2.10. Pantalla del caso de uso “Variables lingüísticas”*

Como se observa en la imagen de la pantalla para el caso de uso “Definir Variables Lingüísticas” (Figura 2.10), el menú permanece igual a lo largo de toda la aplicación. El área de trabajo cuenta con botones alusivos a los diferentes pasos asociados a la ejecución de cada tarea, así como también con un esquema de trabajo que muestra al usuario un diagrama que ilustra dichos pasos. Se dispone de unos botones de limpieza de datos (el cual permite reiniciar los mismos) y de ayuda. En la parte inferior de la pantalla, se puede observar el progreso de trabajo, el cual muestra el estado de ejecución de cada uno de los pasos asociados a la tarea.

Para empezar a diseñar el sistema de ejemplo, se crean las variables lingüísticas de entrada y salida (Figura 2.11). En esta misma pantalla se puede observar la funcionalidad de edición y eliminación de variables lingüísticas ya creadas:


*Figura 2.11. Pantalla de la funcionalidad "Crear/Editar Variable Lingüística"*

Es importante destacar que en todas las funcionalidades donde se ingresen datos estos son validados para garantizar el correcto estado de los mismos en el sistema.

Cada pantalla de funcionalidad tiene un área de trabajo y otra de descripción, en donde se guía al usuario sobre la tarea que debe realizar.

Para crear una nueva variable, se debe seleccionar la opción "[Crear nueva variable]"; si se desea editarla, se selecciona la variable correspondiente. A continuación se debe seleccionar el tipo de variable, ingresar su nombre y rango y por último hacer clic en el botón "Guardar".

Si se desea eliminar una variable, se debe seleccionar y hacer clic en el botón "Eliminar variable".

Al finalizar la creación de todas las variables lingüísticas del sistema a desarrollar, el área de trabajo cambia para mostrar los datos creados (Figura 2.12):

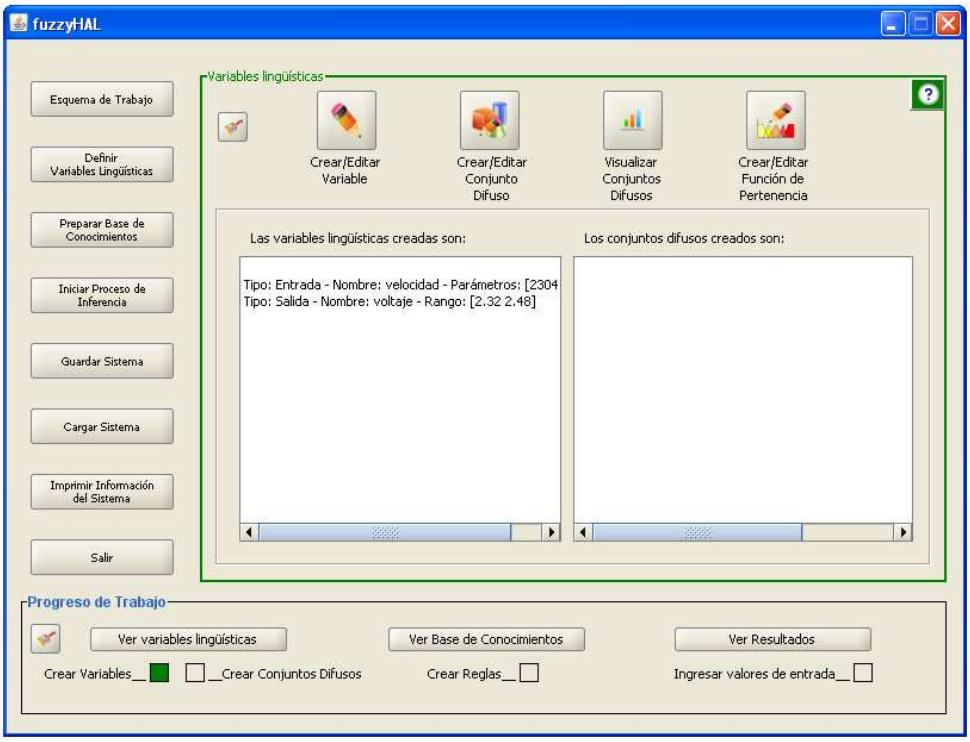



*Figura 2.12. Pantalla del caso de uso "Definir Variables Lingüísticas" con la información actualizada*

El siguiente paso consiste en crear todos los conjuntos difusos asociados a cada variable lingüística definida (Figura 2.13):


*Figura 2.13. Pantalla de la funcionalidad "Crear/Editar Conjunto Difuso"*

En este paso se pueden crear, editar o eliminar los conjuntos difusos de las variables lingüísticas creadas. Se valida que no exista el mismo nombre del conjunto asociado a una

misma variable lingüística, como también que los parámetros definidos para la función de pertenencia seleccionada se encuentren dentro del rango de la variable lingüística asociada.

Para crear un nuevo conjunto difuso se debe seleccionar la opción "[Crear nuevo conjunto difuso]"; si se desea editarlo se selecciona el conjunto difuso correspondiente. A continuación, se ingresa el nombre del conjunto, se selecciona la función de pertenencia y se colocan los parámetros asociados a esta.

Dependiendo de la función de pertenencia seleccionada, se indica como ayuda el número de parámetros asociado, así como también se puede consultar una pantalla que muestra las características de dicha función (Figura 2.14):

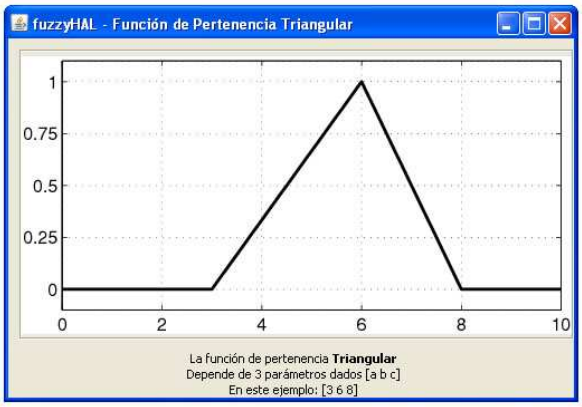



*Figura 2.14. Pantalla que muestra las características de la función de pertenencia "Triangular"*

Al finalizar la creación de los conjuntos difusos correspondientes, el área de trabajo cambia para mostrar los datos creados (Figura 2.15):

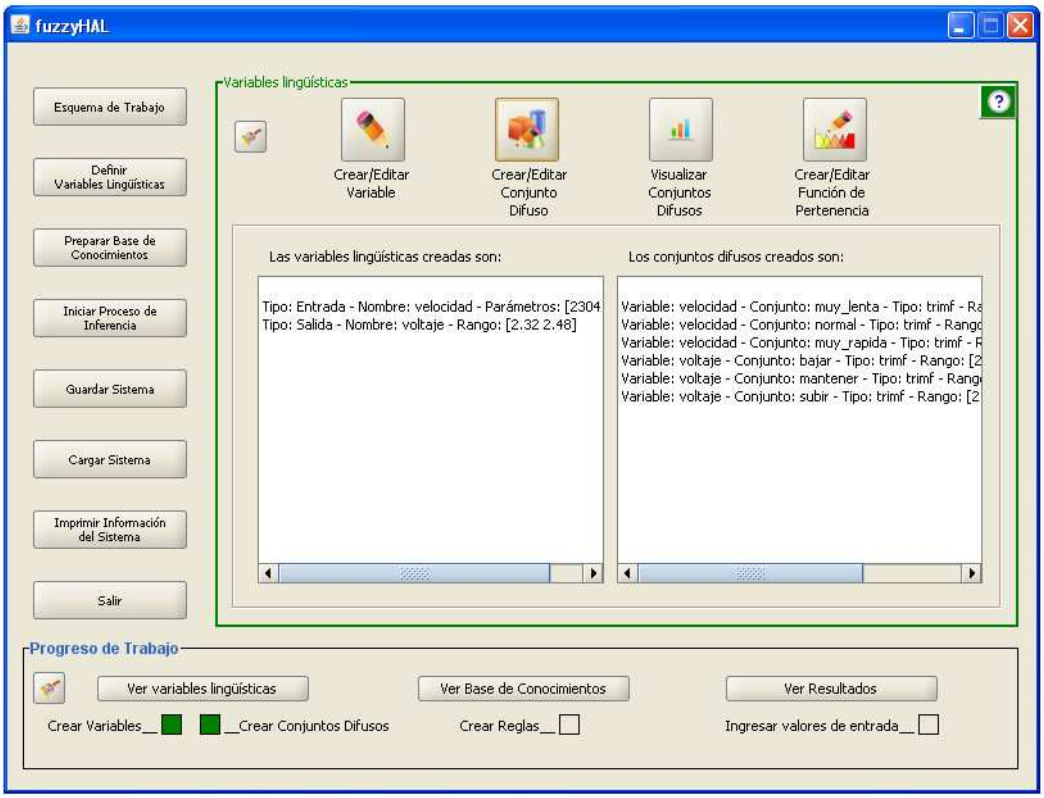



*Figura 2.15. Pantalla del caso de uso "Definir Variables Lingüísticas" con la información actualizada*

A medida que se vayan creando los conjuntos difusos, se pueden ir visualizando en la siguiente pantalla (Figura 2.16):

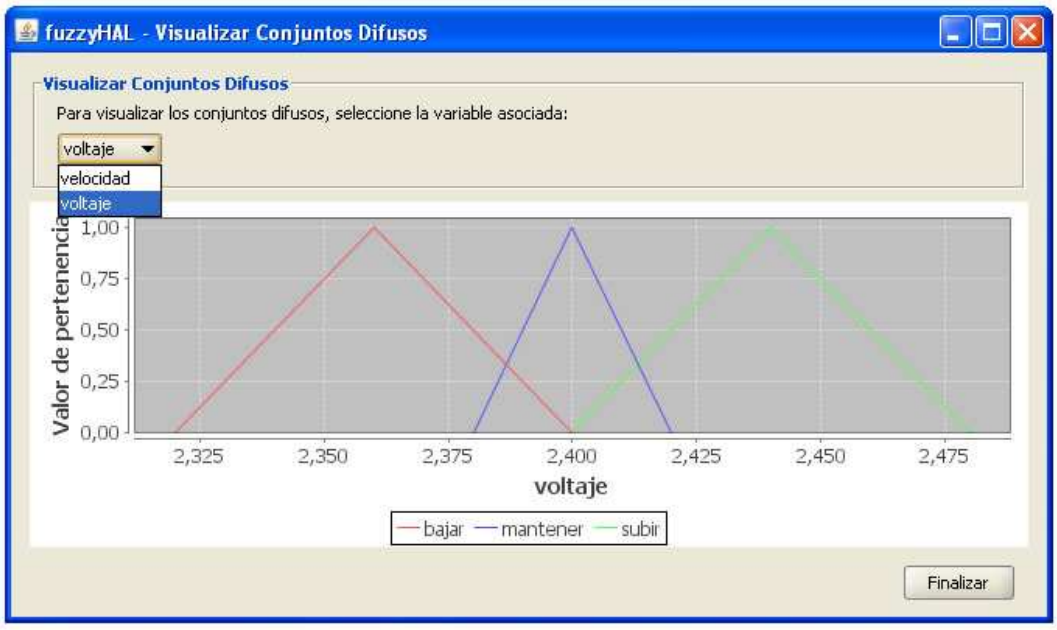



*Figura 2.16. Pantalla de visualización de los conjuntos difusos creados*

Adicionalmente, en esta sección se pueden definir o editar funciones de pertenencia creadas por el usuario, haciendo clic en el botón “Crear/Editar Función de Pertenencia”. Se debe colocar el nombre de la función, el número de parámetros que ésta posee y por último, se coloca la definición de la función, siguiendo la sintaxis del lenguaje de programación JAVA (Figura 2.17):

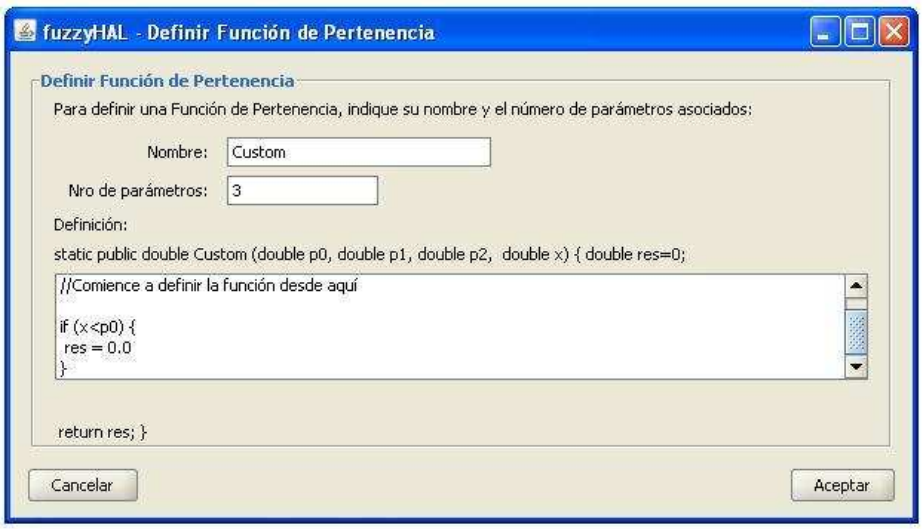



*Figura 2.17. Pantalla de la funcionalidad “Crear/Editar función de pertenencia”*

Al concluir con la creación o edición de la función, el sistema hace una compilación en tiempo real del código definido y en caso de conseguir errores, estos son mostrados al usuario para que conozca su localización para su posterior corrección (Figura 2.18):


*Figura 2.18. Información de los errores de compilación detectados*

Una vez que han sido creadas las variables lingüísticas y sus conjuntos difusos, el próximo paso es crear la Base de Conocimientos.

2. Definición de la Base de Conocimientos

Para definir la Base de Conocimientos se ingresa en el área del Caso de Uso “Preparar Base de Conocimientos”, cuya pantalla se muestra a continuación (Figura 2.19):

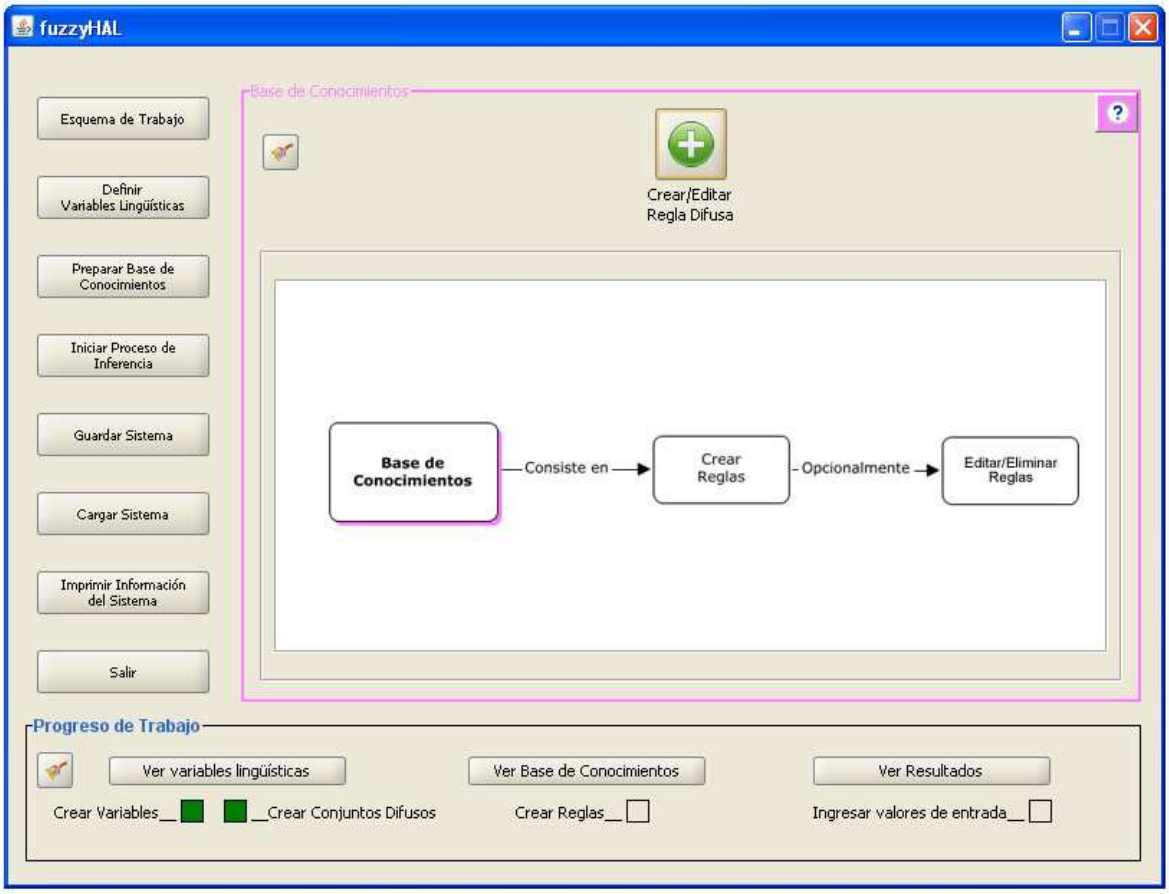



*Figura 2.19. Pantalla del caso de uso “Preparar Base de Conocimientos”*

Al hacer clic en el botón “Crear/Editar Regla Difusa” se muestra la siguiente pantalla (Figura 2.20), donde se pueden realizar las operaciones de creación, edición y eliminación de reglas difusas:

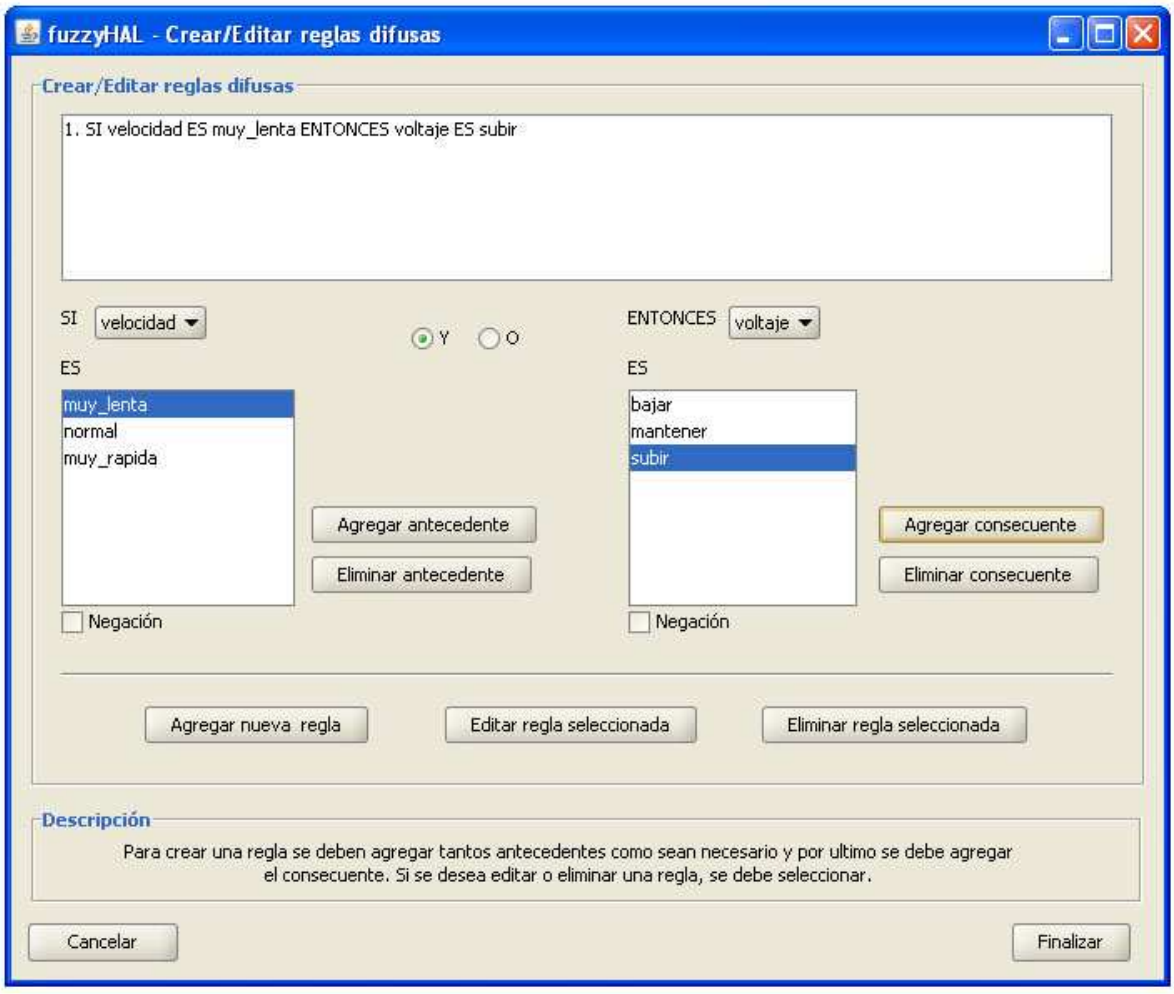



*Figura 2.20. Pantalla de la funcionalidad "Crear/Editar reglas difusas"*

Para crear una regla, se deben crear tantos antecedentes como sean necesarios; para esto se selecciona la variable lingüística de entrada con su conjunto difuso correspondiente, en caso de que se quiera colocar la negación del conjunto difuso seleccionado, se debe elegir la casilla "Negación". A continuación se hace clic en el botón "Agregar antecedente".

Si se desea agregar un nuevo antecedente, se selecciona el operador lógico correspondiente ("Y" u "O") y se repite el proceso descrito anteriormente. A continuación se debe agregar de la misma forma el consecuente de la regla.

Si se desea agregar una nueva regla, se debe hacer clic en el botón "Agregar nueva regla", en caso contrario, se debe hacer clic en el botón "Finalizar".

Si se desea editar una regla dada, se selecciona en el listado de reglas creadas y se modifican sus valores siguiendo los mismos pasos descritos para la definición de las reglas y al finalizar se hace clic en el botón “Editar regla seleccionada”.

Si se desea eliminar una regla dada, se selecciona en el listado de reglas creadas y se hace clic en el botón “Eliminar regla seleccionada”.

Al finalizar la creación de todas las reglas difusas del sistema a desarrollar, el área de trabajo cambia para mostrar las reglas creadas (Figura 2.21):

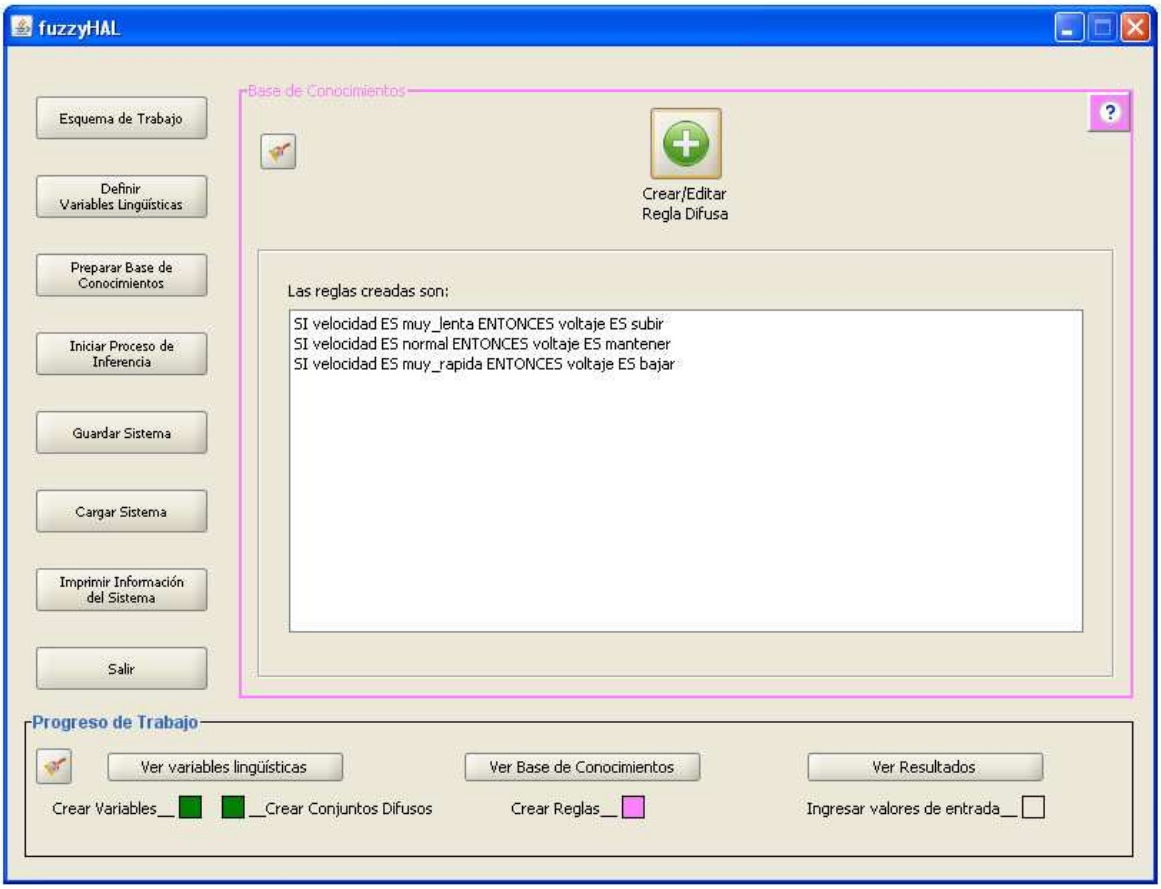



*Figura 2.21. Pantalla del caso de uso “Preparar Base de Conocimientos” con la información actualizada*

Una vez que han sido creadas las variables lingüísticas, los conjuntos difusos y reglas difusas, el próximo paso es iniciar el Proceso de Inferencia.

3. Iniciar Proceso de Inferencia

A continuación se ingresa en el área del Caso de Uso “Iniciar Proceso de Inferencia”, cuya pantalla se muestra a continuación (Figura 2.22):

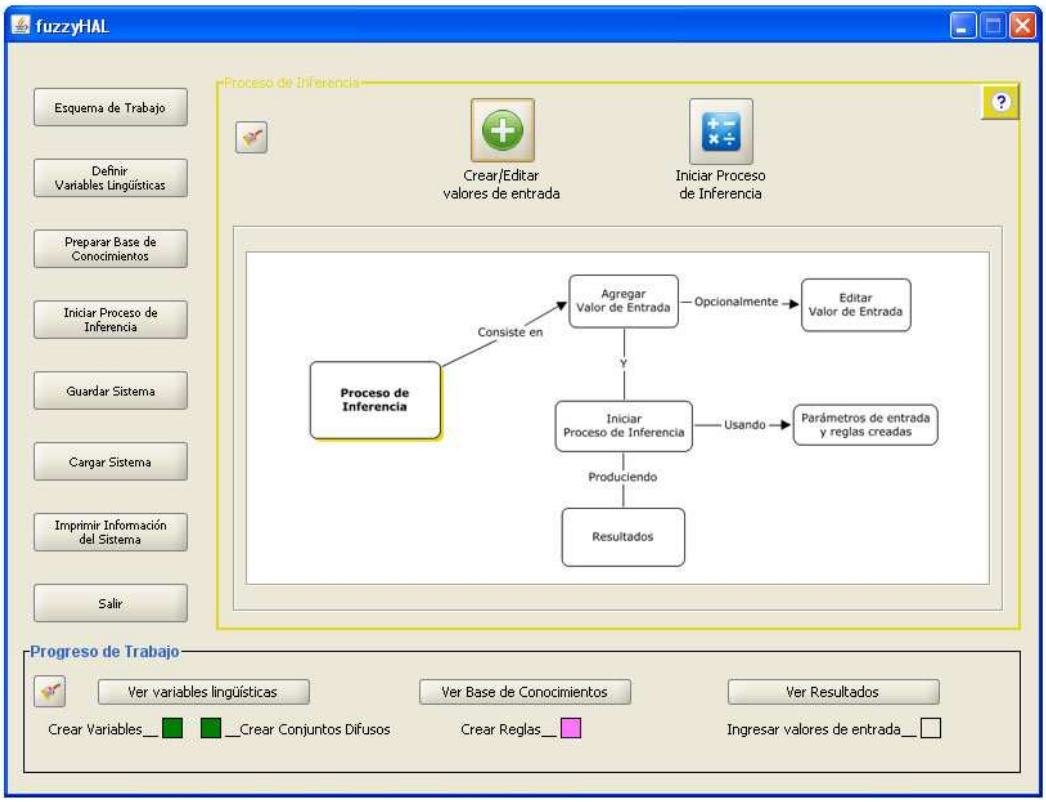



*Figura 2.22. Pantalla del caso de uso “Iniciar Proceso de Inferencia”*

Al hacer clic en el botón “Crear/Editar valores de entrada” se muestra la siguiente pantalla (Figura 2.23) donde se cuenta con la funcionalidad de creación y edición de los valores asociados a las variables de entrada al sistema:


*Figura 2.23. Pantalla de la funcionalidad "Crear/Editar valores de entrada"*

Para introducir o editar un valor a una variable de entrada, ésta debe ser seleccionada y luego se ingresa el valor. Para el ejemplo descrito se va a indicar el valor de 2322 a la variable de entrada "velocidad".

Al finalizar la creación de los valores de entrada, el área de trabajo cambia para mostrar los datos creados, así como también los métodos que se deben seleccionar para ser utilizados en el proceso de inferencia (Figura 2.24):

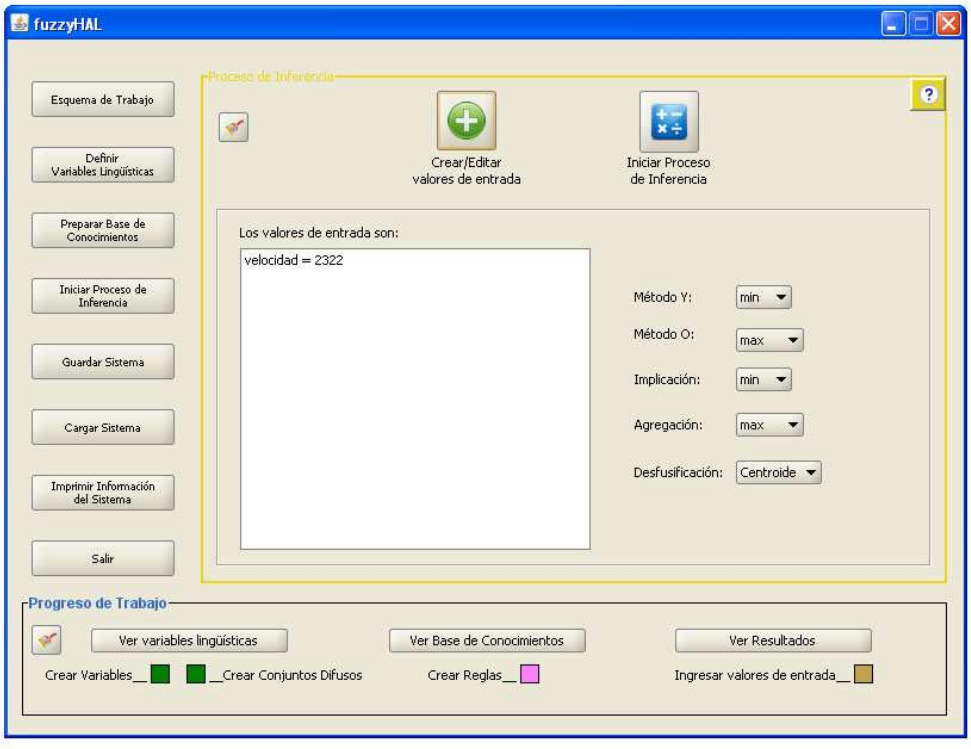



*Figura 2.24. Pantalla del caso de uso "Iniciar Proceso de Inferencia" con la información actualizada*

Como último paso se deben seleccionar los métodos a utilizar en el proceso de inferencia, en este caso se seleccionaron los indicados para el ejemplo a desarrollar, en la Tabla 2.31.

A continuación, se hace clic en el botón “Iniciar Proceso de Inferencia” para obtener los resultados finales. Luego de que el sistema calcula el resultado, se muestra una pantalla donde se indican las reglas que han sido disparadas y se muestra gráficamente el resultado de la agregación de los conjuntos resultantes y el valor de salida desfusificado (Figura 2.25), culminando de esta manera el proceso de creación del sistema basado en reglas difusas. El resultado para la variable de salida “voltaje” es 2,44.

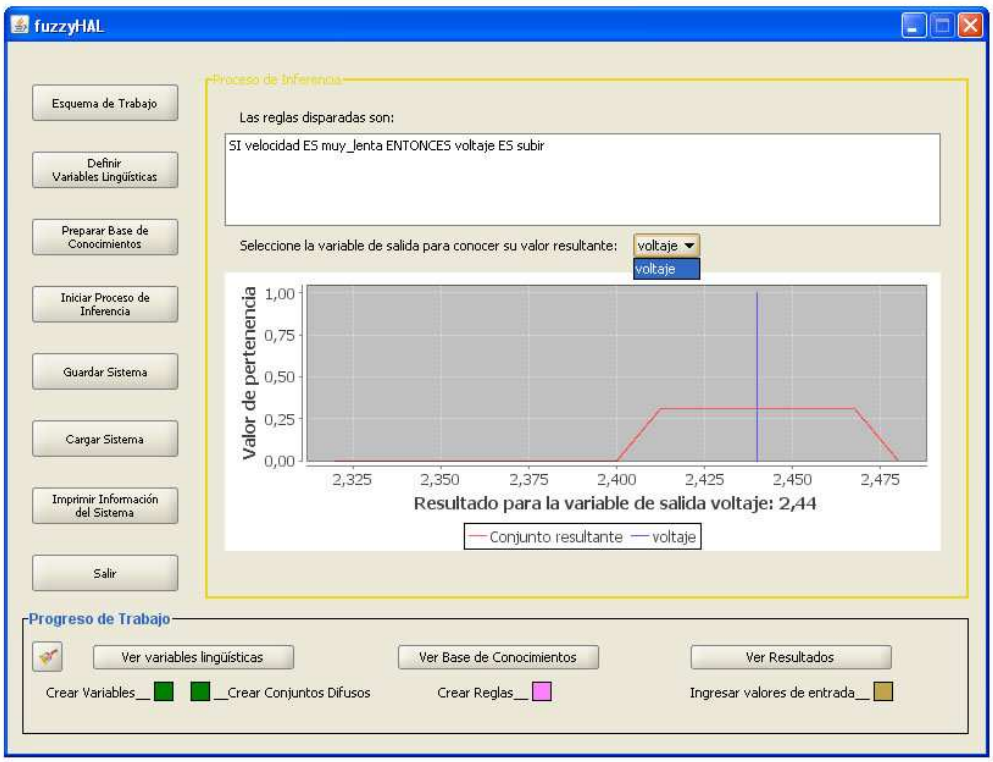
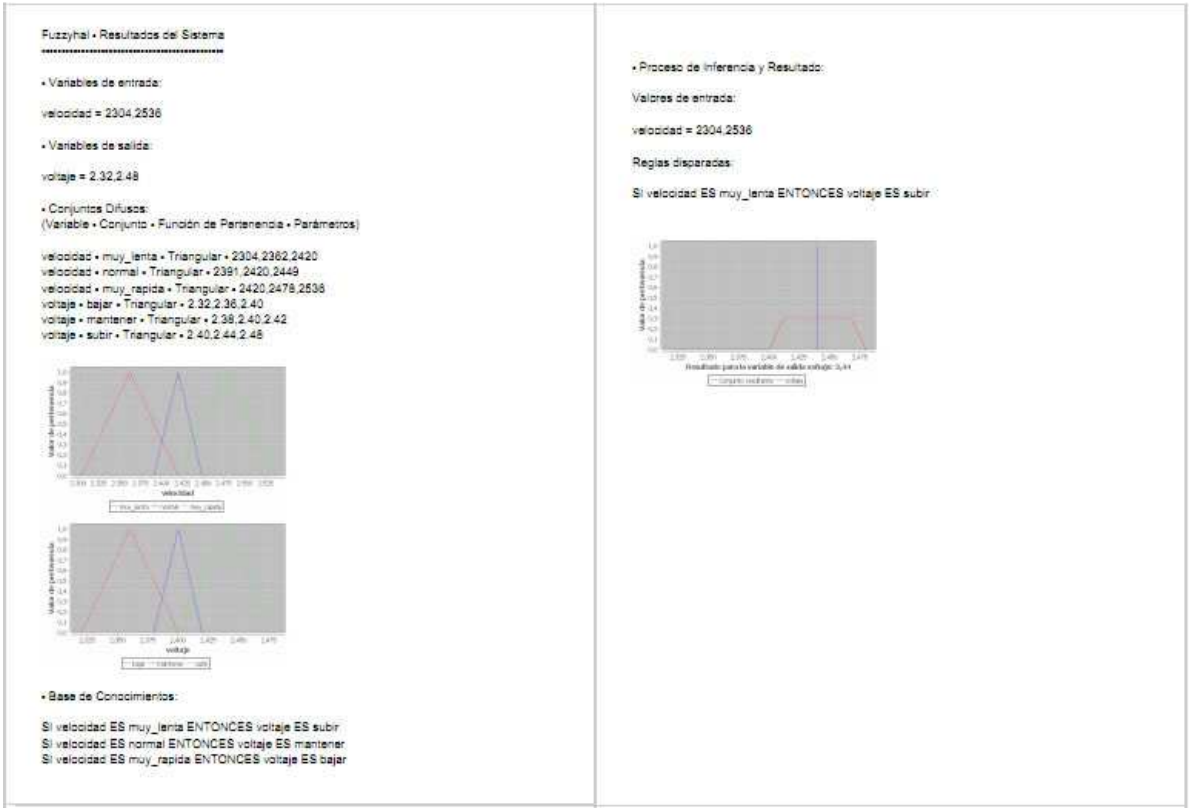



*Figura 2.25. Pantalla que muestra los resultados arrojados por el proceso de inferencia*

Adicionalmente es posible almacenar el sistema creado para su posterior recuperación (Figura 2.26), estos archivos se almacenan con la extensión “.fs”. También es posible imprimir documentación del mismo donde se describen todas sus características. En la Figura 2.27 se puede ver el contenido de dicho documento.


*Figura 2.26. Pantalla de la funcionalidad "Guardar Sistema"*


Fuzzyhal - Resultados del Sistema

- Variables de entrada:

velocidad = 2304.2536

- Variables de salida:

voltaje = 2.32,2.48

- Conjuntos Difusos:
(Variable - Conjunto - Función de Pertenencia - Parámetros)

velocidad - muy_lenta - Triangular - 2304.2362,2420
velocidad - normal - Triangular - 2391,2420,2449
velocidad - muy_rapida - Triangular - 2420,2478,2536
voltaje - bajar - Triangular - 2.32,2.36,2.40
voltaje - mantener - Triangular - 2.38,2.40,2.42
voltaje - subir - Triangular - 2.40,2.44,2.48

- Base de Conocimientos:

SI velocidad ES muy_lenta ENTONCES voltaje ES subir
SI velocidad ES normal ENTONCES voltaje ES mantener
SI velocidad ES muy_rapida ENTONCES voltaje ES bajar

- Proceso de Inferencia y Resultado:

Valores de entrada:

velocidad = 2304.2536

Reglas disparadas:

SI velocidad ES muy_lenta ENTONCES voltaje ES subir

*Figura 2.27. Documentación generada con los datos del sistema creado*

## 2.4 Experimentos y resultados

Para comprobar el funcionamiento de la herramienta y la validez de sus resultados se hicieron pruebas comparativas con el Toolbox de Lógica Difusa incluído en la herramienta MATLAB. La característica a comparar fue el resultado obtenido al finalizar el proceso de inferencia de un sistema basado en reglas difusas dado.

El sistema que se utilizó para las pruebas es un sistema de control de velocidad de un motor, el cual fue utilizado como Aplicación Ejemplo descrita en la sección anterior.

Luego de definirse el sistema tanto en MATLAB como en fuzzyHAL, se realizaron distintas pruebas con diferentes valores para la variable de entrada y diferentes valores para los distintos métodos asociados al proceso de inferencia; se registraron los resultados dados por ambas herramientas (Tabla 2.33), los cuales fueron iguales, siendo satisfactorios los resultados de la aplicación de la prueba comparativa.

*Tabla 2.33. Resultados dados por ambas herramientas*

| | Prueba #1 | Prueba #2 | Prueba #3 | Prueba #4 | Prueba #5 |
|---|---|---|---|---|---|
| Valor de entrada "velocidad" | 2322 | 2415 | 2432 | 2443 | 2532 |
| Método Y | min | min | min | prod | min |
| Método O | max | max | probor | probor | probor |
| Método Implicación | min | prod | min | prod | min |
| Método Agregación | max | sum | probor | probor | probor |
| Método Desfusificación | Centroide | Centroide | mom | som | mom |
| **Resultado MATLAB "voltaje"** | **2.44** | **2.41** | **2.39** | **2.36** | **2.36** |
| **Resultado fuzzyHAL "voltaje"** | **2.44** | **2.407** | **2.392** | **2.36** | **2.36** |

En las siguientes Figuras (Figuras 2.28 y 2.29) se muestran los resultados para la Prueba #1 dados por MATLAB y fuzzyHAL respectivamente:

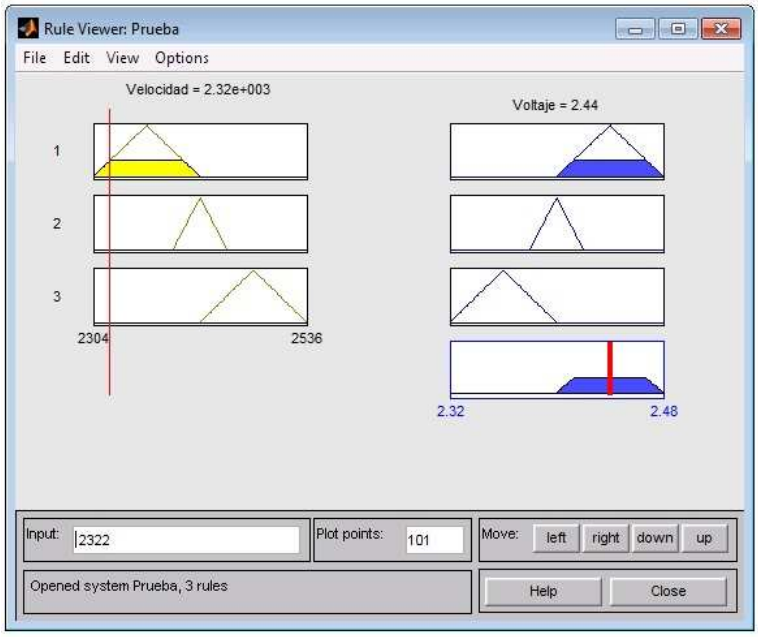



*Figura 2.28. Resultado para la Prueba #1 dado por MATLAB*

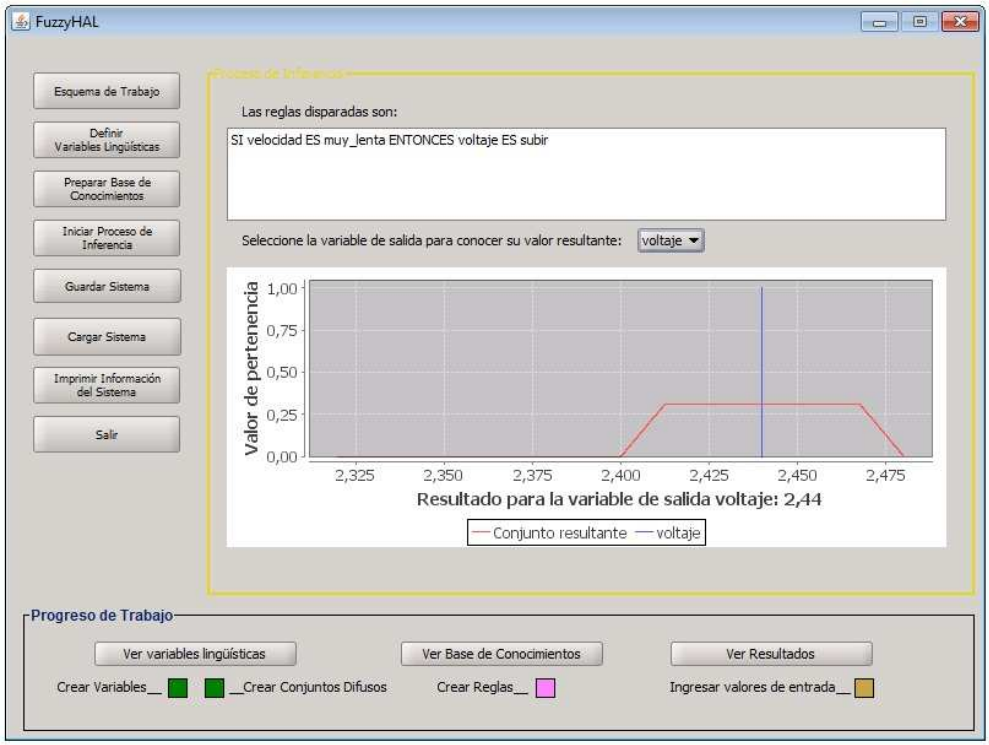



*Figura 2.29. Resultado para la Prueba #1 dado por fuzzyHAL*

En las siguientes Figuras (Figuras 2.30 y 2.31) se muestran los resultados para la Prueba #2 dados por MATLAB y fuzzyHAL respectivamente:

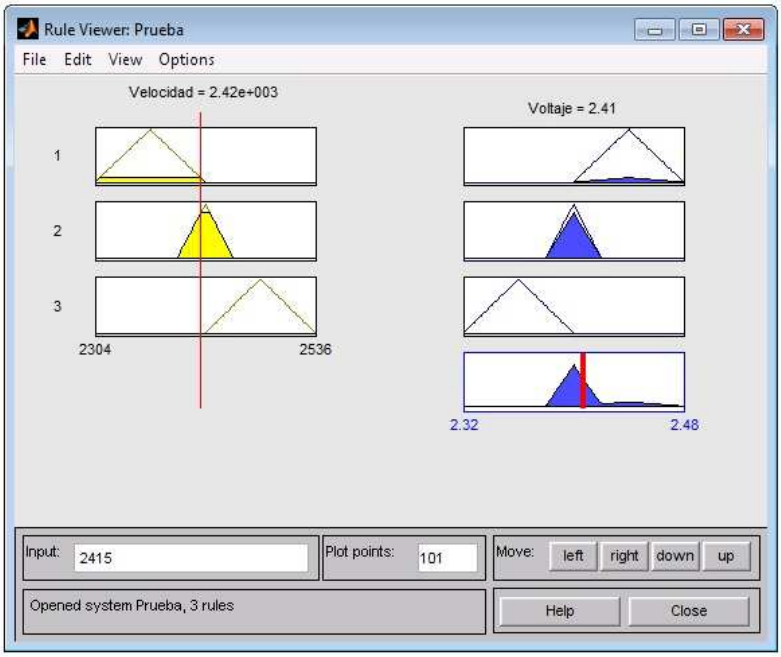



*Figura 2.30. Resultado para la Prueba #2 dado por MATLAB*

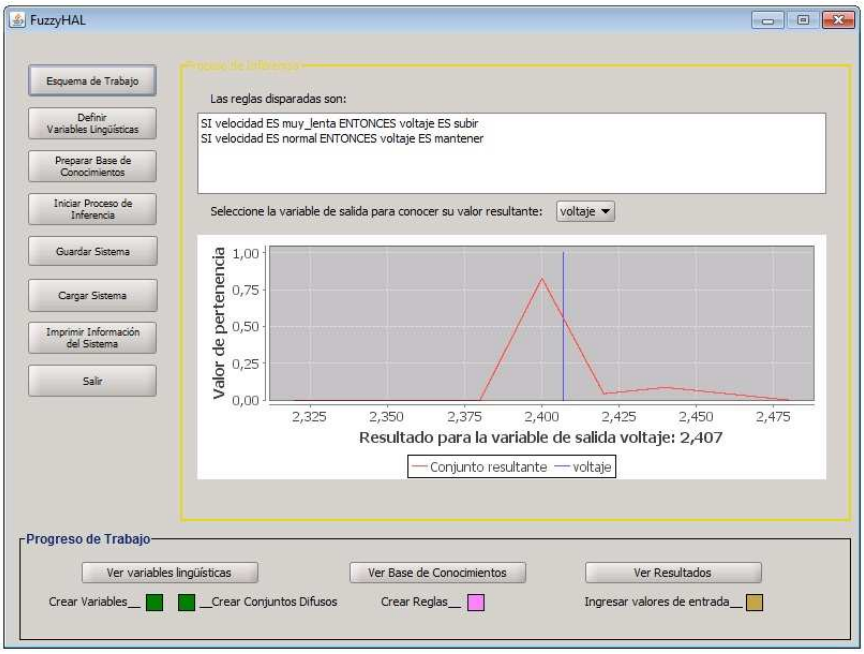



*Figura 2.31. Resultado para la Prueba #2 dado por fuzzyHAL*

En las siguientes Figuras (Figuras 2.32 y 2.33) se muestran los resultados para la Prueba #3 dados por MATLAB y fuzzyHAL respectivamente:

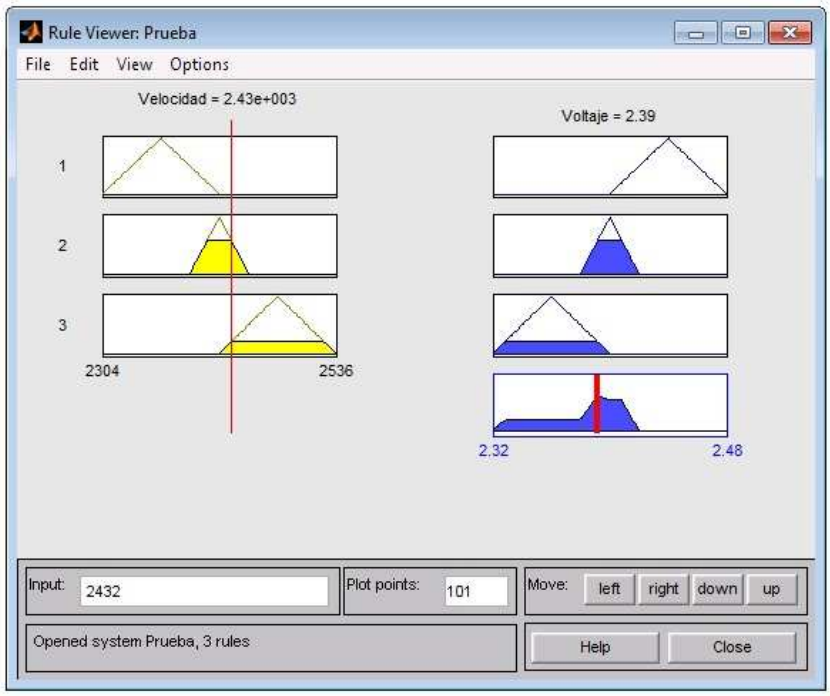



*Figura 2.32. Resultado para la Prueba #3 dado por MATLAB*

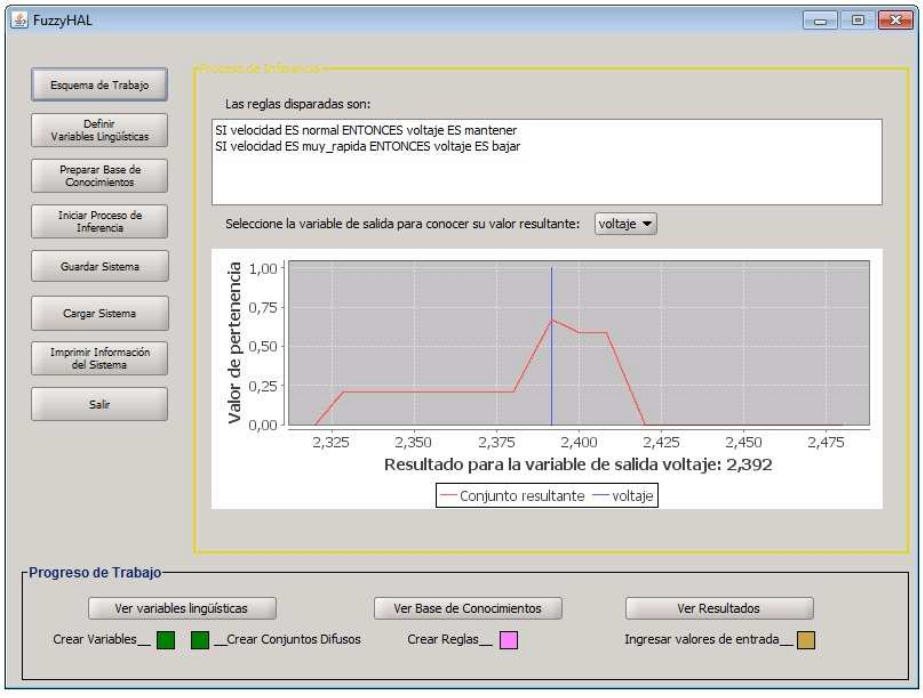



*Figura 2.33. Resultado para la Prueba #3 dado por fuzzyHAL*

En las siguientes Figuras (Figuras 2.34 y 2.35) se muestran los resultados para la Prueba #4 dados por MATLAB y fuzzyHAL respectivamente:

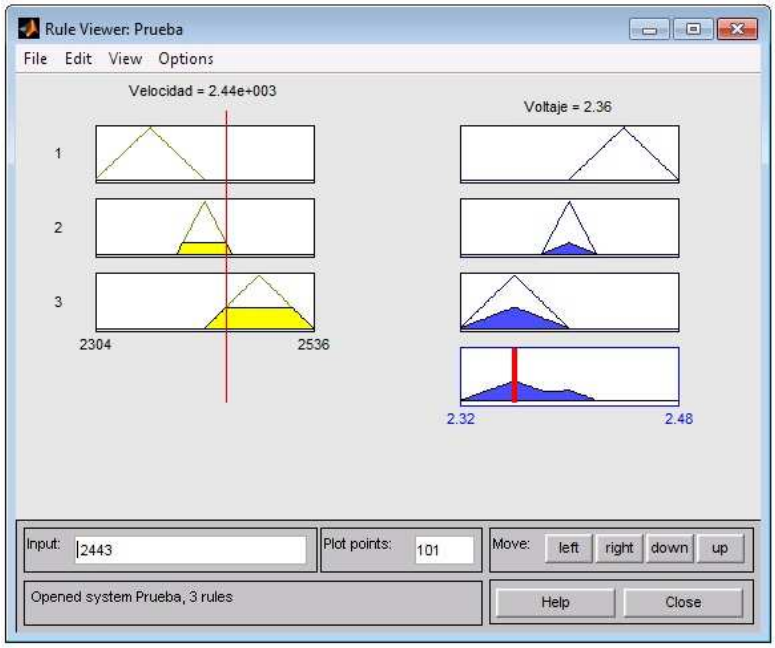



*Figura 2.34. Resultado para la Prueba #4 dado por MATLAB*

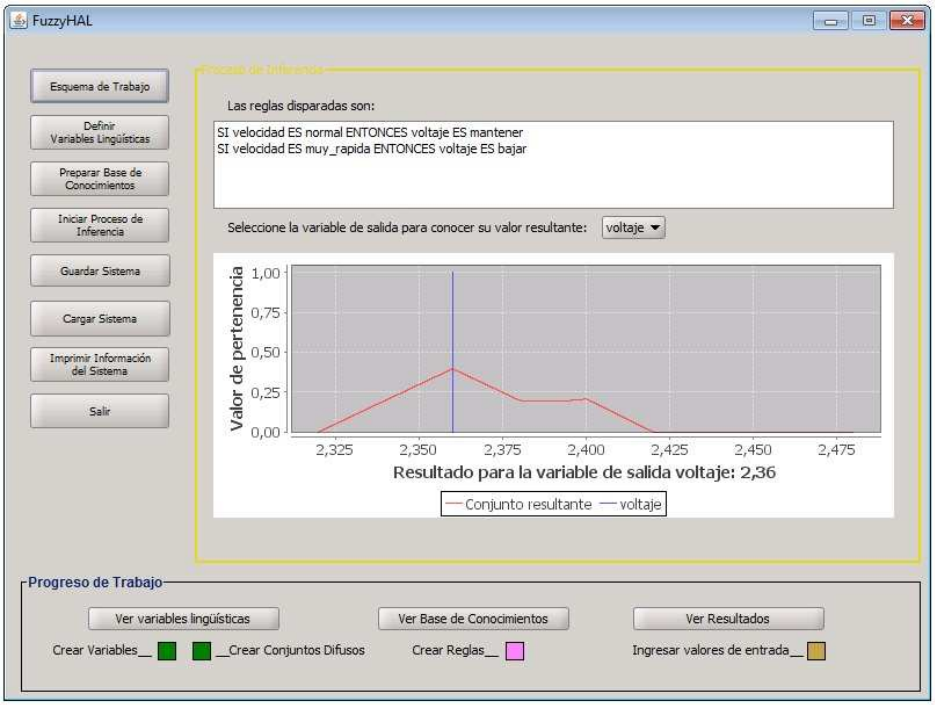



*Figura 2.35. Resultado para la Prueba #4 dado por fuzzyHAL*

En las siguientes Figuras (Figuras 2.36 y 2.37) se muestran los resultados para la Prueba #5 dados por MATLAB y fuzzyHAL respectivamente:

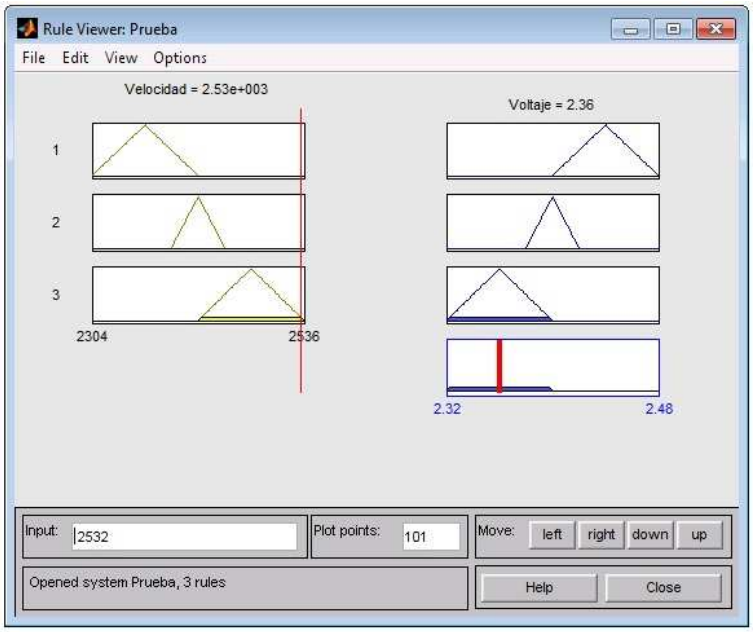



*Figura 2.36. Resultado para la Prueba #5 dado por MATLAB*

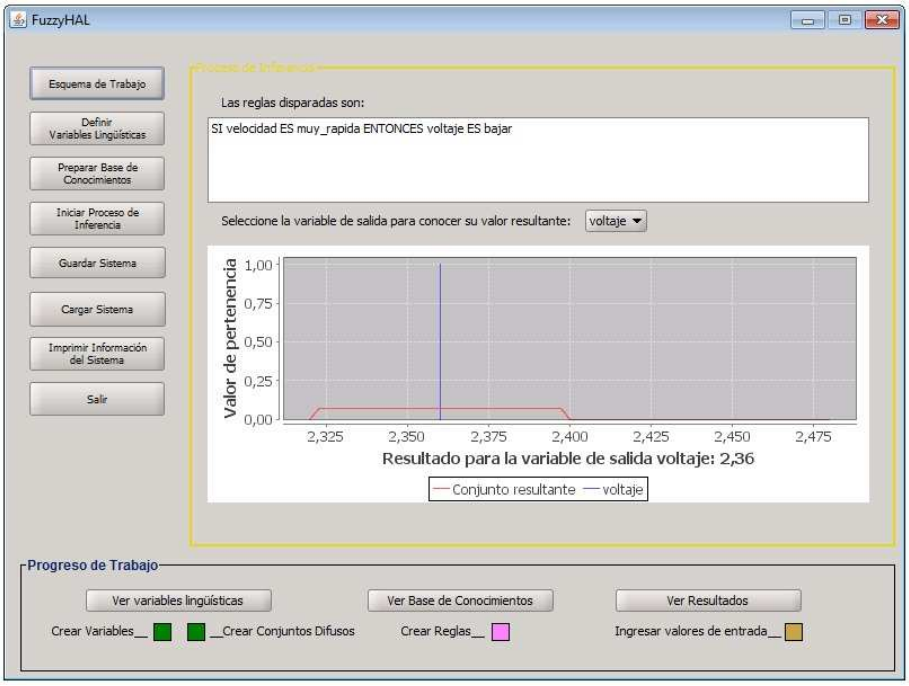



*Figura 2.37. Resultado para la Prueba #5 dado por fuzzyHAL*

# CONCLUSIONES

En la actualidad existen varias herramientas para desarrollar sistemas basados en reglas difusas, sin embargo, la mayoría no son de libre distribución o su usabilidad es limitada; por esta razón, se propuso como solución el desarrollo de un prototipo de herramienta con el mismo propósito, la cual está destinada tanto a usuarios que dominan el tema de la lógica difusa como a principiantes y además es de libre distribución. Luego de haber estudiado distintas herramientas existentes en el mercado, se lograron reunir las mejores características y funcionalidades en un ambiente cuya interfaz es intuitiva y de fácil uso.

Dado que la herramienta es de libre distribución, se permite que tanto estudiantes como expertos en el tema puedan tener fácil acceso a una aplicación que les permita diseñar sistemas basados en reglas difusas, contando con todas las funcionalidades necesarias y un ambiente de óptima usabilidad, permitiendo así estudiar e implementar el uso de sistemas expertos en áreas como la medicina, el control industrial, meteorología, tecnología, educación, etc.

Fue considerada de suma importancia la interacción con la herramienta mediante la aplicación de una interfaz gráfica intuitiva que permitiera el desarrollo de un sistema basado en reglas difusas de manera sencilla, guiando al usuario a través de todo el proceso. Se aseguró el cumplimiento de este objetivo al basar la interfaz en los lineamientos planteados por el Lic. Gustavo Torres, para su Trabajo Especial de Grado llamado "*Integración y Mejoramiento Continuo a la Herramienta de Redes Neuronales del Laboratorio de Inteligencia Artificial*" (Torres, 2006), donde fueron aplicadas distintas pruebas de usabilidad para garantizar una óptima calidad en la interacción entre el usuario y la herramienta.

Al culminar el desarrollo, se estableció una prueba de control usando el Toolbox de Lógica Difusa de MATLAB (MATLAB, 2009) de referencia, donde se corroboró la veracidad de los resultados arrojados por la aplicación. En esta prueba de control se diseñó el mismo sistema basado en reglas difusas en ambas aplicaciones.

Adicionalmente, la aplicación podrá ser descargada en el sitio web del Laboratorio de Inteligencia Artificial del Centro de Ingeniería de Software y Sistemas, Facultad de Ciencias, Universidad Central de Venezuela (http://lia.ciens.ucv.ve), facilitando de esta manera la libre distribución de la herramienta.

# REFERENCIAS


- Araújo, L., Lizárraga, M., Sucupira, L., Yabu-uti, J. y Ling, L. (2004). Autentificación personal por dinámica de tecleo basada en lógica difusa. *IEEE Latin American Transactions 2* (1), 69-75.

- Black, M. (1937). Vagueness: an exercise in logical analysis. *Philos. Soc 4* (1937), 427-455.

- Escobar, M., Sierra, E., Lajes, S. (2004). Determinación del nivel de deterioro en líneas eléctricas, *Ingenierías 7* (23)*,* 30-36.

- Electrical Engineering and Computer Sciences Department, Berkeley, University of California. *FLINT Fuzzy Logic and the Internet,* 2009, [en línea]. Disponible: http://www-bisc.eecs.berkeley.edu/FLINT/

- Ferreira, A. y Franklin, C. (2007). Fuzzy inference to risk assessment on nuclear engineering Systems. *Applied Soft Computing 7.* 17-28.

- Goncalves, M., Tineo, L., León, G. y Martínez, D. (2002). Una herramienta en web para la evaluación de desempeño docente, sobre un sistema de consultas difusas. *Acta Científica Venezolana 53 (1).* 365-366.

- González-Morcillo, C, Jiménez, L., Moreno-García, J. Aplicación de Lógica Difusa en Sistemas de Captura de Movimiento, *Actas del XII Congreso Español Sobre Tecnologías y Lógica Fuzzy, Estylf'04*, 27-32.

- Gupta, M. y Yamakawa, T. (1988). *Fuzzy logic in knowledge-based systems, decision and control.* New York: Elsevier Science Inc.

- IEC (2000). *IEC 61131-7: Programmable controllers - Part 7: Fuzzy control programming*. International Electrotechnical Commission. [en línea]. Disponible: http://www.iec.ch/

- Inform. *FuzzyTECH 5.72,* 2009, [en línea]. Disponible: http://www.fuzzytech.com/

- Instituto de Microelectrónica de Sevilla. Lenguaje XFL, 2001, [en línea]. Disponible: http://www.imse-cnm.csic.es/xfl/

- Instituto de Microelectrónica de Sevilla. XFuzzy 2.1, 2001, [en línea]. Disponible: http://www.imse.cnm.es/Xfuzzy/Xfuzzy_2.1/index_sp.htm

- JGoodies Looks R1.3.2, 2005, [en línea]. Disponible: http://www.jgoodies.com/downloads/libraries.html

- Kosko, Bart. (1999). *The fuzzy future: from society and science to heaven in a chip*. Harmony Books.

- Li, Zhong. (2006). *Fuzzy Chaotic Systems: Modeling, Control, and Applications*. New York: Springer.

- Łukasiewicz, J. (1920), O logice trojwartosciowej, *Ruch Filozoficny*, 5, 170–171.

- Mendel, Jerry. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and new directions*. New Jersey: Prentice-Hall.

- Nasser, S., Vert, G.L., Nicolescu, M. y Murray, A. (2007) Multiple Sequence Alignment using Fuzzy Logic, *Computational Intelligence and Bioinformatics and Computational Biology.*

- Ngai, E., Wat, F. (2003). Design and development of a fuzzy expert system for hotel selection. *OMEGA The international journal of Managment Science 31.*275-286.

- Nguyen, H.T. y Walker, E.A. (1996). *A first course in Fuzzy Logic*. Boca Ratón, Florida: CRC Press.

- NRC Institute for Information Technology. FuzzyCLIPS 6.10c, 2009, [en línea]. Disponible: http://www.nrc-cnrc.gc.ca/IR_public/fuzzy/fuzzyClips/

- Novak, V. (1987). First-order fuzzy logic. *Studia Logica Library 46*, 87-109.

- Pedrycz, W. (1995). *Fuzzy Sets Engineering*. Boca Ratón, Florida: CRC Press.

- Pimentel, Geraldine (2008). *Herramienta de minería de datos: prototipo del módulo de extracción de reglas de clasificación y reglas de asociación.* Trabajo Especial de Grado. Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela.

- Ponce, Sergio (2001). Incubadora de cuidados intensivos controlada con lógica difusa. *Memorias II Congreso Latinoamericano de Ingeniería Biomédica.*

- Roussel, O., Cavelier, A., y Hayo, M. (2000). Adaptation and use of a fuzzy expert system to asses the envirnmental effect of pesticides applied to field crops. *Agriculture, Ecosystems and Environment 80.* 143-158.

- Saritas, I., Allahverdi, N., Serti, I. (2003). A Fuzzy Expert System Design for Diagnosis of Prostate Cancer, *International Conference on Computer Systems and Technologies - CompSysTech'2003.*

- Schneider, Moti; Kandel, Abraham; Langholz, Gideon y Chew, Gerard. (1996). *Fuzzy Expert System Tools*. West Sussex, England: John Wiley & Sons.

- The MathWorks. *MATLAB R2009a,* 2009, [en línea]. Disponible: http://www.mathworks.com/products/matlab/

- Tineo, L. (2002). Sistemas de bases de datos orientadas a objetos difusos. *Acta Científica Venezolana 53 (1).* 364.

- Torres, Gustavo (2006). *Integración y Mejoramiento Continuo a la Herramienta de Redes Neuronales del Laboratorio de Inteligencia Artificial.* Trabajo Especial de Grado. Escuela de Computación. Universidad Central de Venezuela, 2006.

- Zadeh, L. (1965). Fuzzy sets, *Information Control 8* (3), 338-353.

- Zadeh, L. (1979). A theory of approximate reasoning. *Machine Intelligence 9.* 149-194.

- Zambrano, Adriana (2008). *Herramienta de minería de datos: prototipo del módulo de extracción de árboles de decisión.* Trabajo Especial de Grado. Escuela de Computación. Facultad de Ciencias. Universidad Central de Venezuela.