ENTORNO CAD PARA LA ENSEÑANZA DE LA LÓGICA DIFUSA Y SUS APLICACIONES

F. J. Moreno Velo, A. Barriga, S. Sánchez-Solano, I. Baturone

Instituto de Microelectrónica de Sevilla (CNM-CSIC)
Avda. Reina Mercedes s/n, Edif. CICA. 41012-Sevilla. España
velo@imse.cnm.es

RESUMEN

En esta comunicación se describe una herramienta CAD de ayuda a la enseñanza de los conceptos teóricos y prácticos de la lógica difusa, la especificación y ajuste de estos sistemas y sus aplicaciones. El entorno está compuesto por un amplio conjunto de herramientas que cubren las diferentes etapas del diseño de los sistemas difusos: descripción, simulación, aprendizaje y síntesis. Estas herramientas comparten una descripción común en un lenguaje de especificación formal llamado XFL3. El alumno puede definir tanto la estructura del sistema como las funciones requeridas para realizar el razonamiento aproximado y observar los resultados derivados de dichas definiciones. Todas las herramientas del entorno han sido programadas en Java por lo que puede ejecutarse en cualquier sistema operativo.

1. INTRODUCCIÓN

A medida que la aplicación de los sistemas difusos se ha ido generalizando, la enseñanza de esta materia requiere la disponibilidad de herramientas informáticas que faciliten su docencia. En la actualidad existen herramientas, tanto comerciales como universitarias, orientadas al diseño de sistemas difusos [1]. Todos estos entornos suelen desarrollar interfaces gráficas que facilitan la descripción y verificación de los sistemas difusos. Sin embargo, otros aspectos como la obtención de conocimiento, el ajuste automático, la generación de descripciones en diferentes lenguajes de programación o la síntesis hardware suelen estar poco desarrollados en estas herramientas. Otros programas están centrados en desarrollar alguno de estos aspectos, pero presentan serias restricciones en cuanto a los sistemas difusos utilizados y resultan insuficientes como herramientas de desarrollo de sistemas.

Una de las líneas de trabajo de nuestro grupo de investigación consiste en el desarrollo de herramientas CAD de libre distribución, aplicadas al diseño de sistemas difusos, y de forma específica orientada a la docencia. En este sentido hemos desarrollado un software que pretende cubrir todos los aspectos del diseño, incluida la aplicación de técnicas de ajuste automático y la generación de realizaciones hardware de los sistemas en diseñados. Esta línea nos condujo al desarrollo del entorno Xfuzzy 2.0 [2,3], que integra diferentes herramientas de edición, ajuste, verificación y síntesis de sistemas difusos. Todas las herramientas comparten un lenguaje de especificación formal, llamado XFL [4], que permite la descripción de sistemas difusos complejos. También se ha elaborado un conjunto de prácticas de laboratorio en aplicaciones virtuales de control y aproximación de funciones [5]. Todo este entorno docente se ha completado con un libro electrónico que da soporte al aprendizaje y estructura la docencia teórico-práctica. [6].

La nueva versión del entorno Xfuzzy incluye la definición de un nuevo lenguaje de especificación formal, más flexible y potente, que incorpora la posibilidad de utilizar modificadores lingüísticos y de extender el conjunto de funciones utilizadas como conectivas

difusas, modificadores lingüísticos, métodos de concreción y funciones de pertenencia. A partir de este lenguaje se han desarrollado nuevas herramientas que presentan una interfaz gráfica más homogénea y que cubren nuevos aspectos del diseño. El entorno se ha programado íntegramente en lenguaje Java, por lo que puede ser ejecutado sobre cualquier plataforma que incluya el entorno de ejecución Java (JRE). En los siguientes apartados se describen los requisitos que debe incorporar un entorno informático docente y, a continuación se describe brevemente las características de Xfuzzy 3.0.

2. REQUERIMIENTOS DE UN ENTORNO INFORMÁTICO DOCENTE

La lógica difusa es una técnica de razonamiento aproximado adecuada para modelar la incertidumbre presente en el lenguaje natural y emular los mecanismos de razonamiento aproximado utilizados por el cerebro humano. Un sistema difuso expresa el conocimiento en base a un conjunto de reglas de comportamiento que son procesadas de forma numérica. El éxito de este mecanismo de razonamiento ha permitido aplicar esta técnica en campos muy diversos como el control de procesos, procesado de información, minería de datos, sistemas expertos, etc. Es por ello que la docencia de los conceptos de lógica difusa pueden encuadrarse dentro de diversas disciplinas curriculares. La amplitud y extensión de las aplicaciones de la lógica difusa hacen que un entorno informático de apoyo a la docencia deba cubrir diferentes perspectivas: desde los conceptos teóricos del formalismo lógico hasta las aplicaciones concretas en diferentes campos.

Para establecer un marco de referencia útil para analizar los requerimientos de una herramienta de ayuda a la docencia de la lógica difusa vamos a considerar una serie de criterios que permiten evaluar la adecuación de la misma. En este sentido un requisito consiste en el empleo de un lenguaje formal para especificar el sistema. El empleo de un lenguaje formal de especificación resulta la opción más conveniente ya que es legible por los alumnos, lo que puede ayudarle a entender en qué y cómo está trabajando la herramienta con el sistema que está desarrollando. El empleo de un lenguaje formal con una sintaxis y una semántica bien definidas permite verificar propiedades del sistema a partir de su especificación, a la vez que las modificaciones que el sistema pueda sufrir dentro de la herramienta (como, por ejemplo, en el caso de la aplicación de métodos de aprendizaje) pueden ser validadas de manera natural.

Otro aspecto que se requiere hoy en día es disponer de interfaces gráficas de usuario para la definición del sistema. La capacidad de comprender, manipular y analizar la definición de cualquier sistema se ve significativamente aumentada mediante el empleo de técnicas que ofrecen una representación gráfica de su estructura y comportamiento. Por otra parte, el tiempo necesario para el aprendizaje de una determinada herramienta es mucho menor si se dispone de mecanismos intuitivos de acceso.

Desde el punto de vista de comprender los conceptos teóricos de la lógica difusa es conveniente disponer de mecanismos para que puedan ser definidos nuevos operadores directamente por el alumno, permitiendo así la evaluación de nuevos formalismos y técnicas, tanto a nivel de eficiencia algorítmica como de simplicidad en cuanto a la realización final del sistema. Por otro lado también es deseable disponer en el lenguaje de especificación de un conjunto de funciones y operadores predefinidos con objeto de disponer de una funcionalidad básica que permita cubrir la realización de prácticas de laboratorio.

Con objeto de cubrir los diferentes campos de aplicación de los sistemas difusos el entorno de CAD debe incorporar facilidades de depuración, verificación y análisis del sistema que se concretan en herramientas de aprendizaje, simulación y análisis. El empleo de métodos de aprendizaje que faciliten la construcción y ajuste del sistema es una característica

fundamental para una herramienta de desarrollo que pretenda ser útil para sistemas mínimamente complejos. Por su parte, los mecanismos de simulación permiten evaluar el comportamiento algorítmico del sistema. Los mecanismos de análisis engloban al conjunto de procedimientos que permiten obtener información de parámetros relevantes del sistema, bien sea por medio de representaciones (gráficas, numéricas e incluso animaciones en tiempo real) o por medio de emulaciones del sistema en tiempo real. El objetivo de estos mecanismos es proporcionar un conocimiento más directo de cuál es el comportamiento interno del sistema.

Resulta obvio que el objetivo final de una herramienta de CAD debe ser la obtención de una implementación operativa, es decir, la síntesis de una realización del sistema a partir de la especificación que el diseñador ha realizado y ajustado. El método más directo para llevar a cabo la síntesis de una especificación almacenada en una herramienta de CAD es traducir la representación interna que la herramienta contiene a una implementación basada en software.

Nuestro interés se centra en la implementación electrónica de sistemas difusos. Por lo tanto, la herramienta debe generar un circuito basado en una arquitectura específica de motor de inferencia. Dicho circuito se representa mediante el lenguaje de descripción de hardware VHDL y puede ser implementado sobre dispositivos FPGA. De esta forma se facilita el recorrido del flujo de diseño desde la especificación de alto nivel hasta la implementación física del circuito.

3. EL ENTORNO XFUZZY 3.0

Xfuzzy 3.0 [7] es un entorno de desarrollo de sistemas difusos formado por un amplio conjunto de herramientas que pueden aplicarse en las diferentes etapas del proceso de diseño de estos sistemas (Fig. 1). Estas herramientas comparten un mismo lenguaje de especificación formal que describe el sistema difuso en desarrollo.

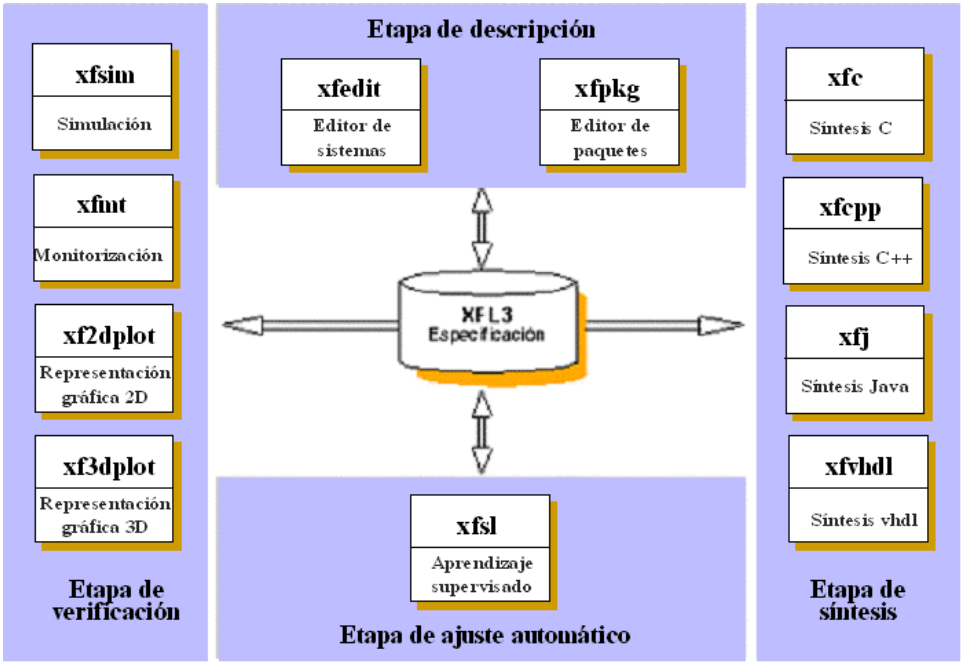

Figura 1. Estructura del entorno Xfuzzy 3.0

La etapa de descripción está formada por herramientas de edición gráfica que permiten describir los sistemas y extender el conjunto de funciones que pueden ser utilizadas en esta descripción. La etapa de ajuste automático está formada por una herramienta de aprendizaje supervisado. Otras herramientas dedicadas a la adquisición de conocimiento y a la

simplificación automática están en estudio para ser incorporadas en esta etapa. La etapa de verificación incluye herramientas para visualizar el comportamiento del sistema. Por último, la etapa de síntesis está formada por herramientas que generan representaciones del sistema en desarrollo en lenguajes de alto nivel. Esta etapa se completa con herramientas que generan realizaciones hardware de los sistemas.

Las diferentes herramientas pueden ser ejecutadas de forma independiente o desde la ventana principal del entorno (Fig. 2). Esta ventana permite trabajar con varios sistemas simultáneamente, contiene enlaces a las diferentes herramientas y permite acceder a la ayuda del entorno.

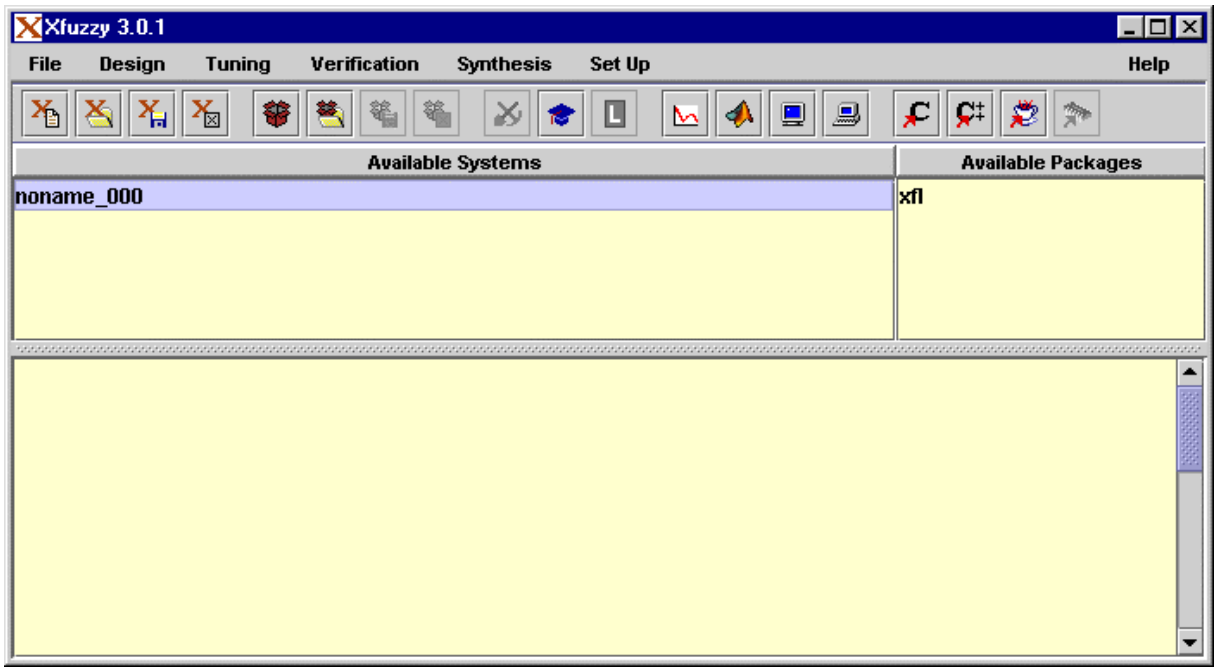

Figura 2. Ventana principal del entorno Xfuzzy 3.0

4. EL LENGUAJE XFL3

XFL3 es un lenguaje de especificación formal que permite describir sistemas difusos [8]. Sus principales características son la separación de la definición de la estructura del sistema respecto a la descripción matemática de los operadores empleados en cada sistema, la capacidad de definir sistemas con bases de reglas jerárquicas, la capacidad de expresar relaciones complejas entre las variables, por medio de diversos modificadores lingüísticos y conectivas difusas, y la posibilidad de ampliar el conjunto de funciones disponibles como conectivas difusas, modificadores lingüísticos, funciones de pertenencia y métodos de concreción.

La estructura lógica del sistema se incluye en ficheros de extensión ".xfl". Estos ficheros están formados por un conjunto de definiciones de objetos, que incluyen conjuntos de operadores, tipos de variables y bases de reglas, y la descripción de la estructura jerárquica de la base de conocimiento del sistema.

Un conjunto de operadores es un objeto que encapsula la asignación de una función matemática a cada operador difuso utilizado en XFL3. Estos operadores son la conjunción y disyunción lógicas, la función de implicación y de agregación de reglas, los modificadores lingüísticos "distinto", "fuertemente", "débilmente" y "ligeramente", y el método de concreción.

Un tipo de variable lingüística es un objeto que describe una variable del sistema. Esta descripción incluye la definición de su universo de discurso, es decir, el rango de definición de la variable, así como la definición de las diferentes etiquetas lingüísticas aplicables a la variable y las funciones de pertenencia asociadas a ellas (Fig. 3).

```
type tipo1 [0,100] {
   bajo xfl.triangle(0,25,50);
   mediano xfl.triangle(25,50,75);
   alto xfl.triangle(50,75,100);
  }

type tipo2 extends tipo1 {
   muy_bajo xfl.triangle(-10,0,25);
   muy_alto  xfl.triangle(75,100,110);
  }
```

Figura 3. Definición de tipos

Una base de reglas consiste en un conjunto de reglas lógicas que relacionan los valores de ciertas variables de entrada con los valores de las variables de salida de la base (Fig. 4). Los antecedentes de estas reglas, que definen la relación entre las variables de entrada, pueden formarse mediante combinaciones de las conectivas difusas *and* y *or* y los modificadores "mayor o igual", "mayor", "menor o igual", "menor", "distinto", "fuertemente", "débilmente" y "ligeramente". En la figura 4 se observa que a las reglas se le puede asociar un factor de certeza (en el caso de la segunda regla toma el valor 0.8) que es un factor multiplicativo del grado de activación de la regla.

```
rulebase base1(tipo1 x, tipo2 y, tipo3 *z) using systemop {
   if( x == mediano & y == mediano) -> z = alto;
   [0.8] if( x<=bajo | y != muy_alto ) -> z = bajo;
   if( +(x>alto) & ~(y==mediano) ) -> z = alto;
   ……..…..  }
```

Figura 4. Ejemplo de base de reglas

La estructura global del sistema describe la jerarquía de razonamiento de las diferentes bases de reglas. La salida de una base de reglas puede utilizarse como entrada de otra base de reglas, formando una serie de razonamientos encadenados que conducen hasta la conclusión final del sistema. Esto permite expresar de forma modular el conocimiento descrito por el sistema difuso.

La descripción de las funciones asociadas a las diferentes operaciones difusas se incluyen en ficheros denominados paquetes, identificados por la extensión ".pkg". Esto incluye la definición matemática de estas funciones en varios lenguajes de programación. Estos paquetes pueden ser creados y ampliados por el usuario.

5. ETAPA DE DESCRIPCIÓN

Xfuzzy 3.0 contiene dos herramientas dedicadas a facilitar la descripción de sistemas difusos, *xfedit* y *xfpkg*, que disponen de interfaces gráficas para describir la estructura lógica del sistema y definir matemáticamente las funciones asociadas a los operadores, respectivamente.

La ventana principal de la herramienta *xfedit* muestra la estructura jerárquica del sistema difuso (Fig. 5). Desde esta ventana es posible definir las variables globales del sistema y los diferentes conjuntos de operadores, tipos de variables y bases de reglas utilizados en la descripción del sistema.

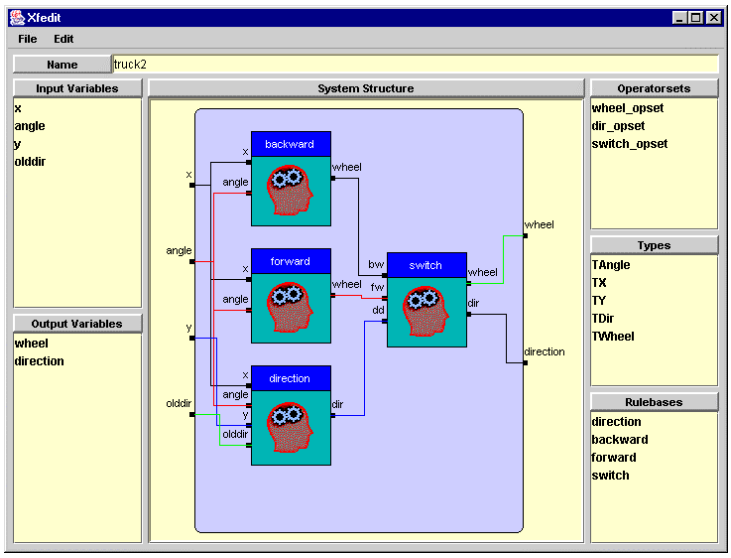

Figura 5. Ventana principal de *xfedit*.

La definición de un conjunto de operadores incluye la introducción del identificador del conjunto y la selección de las funciones asociadas a cada operador.

La ventana de edición de un tipo de variable lingüística incluye una serie de campos para introducir el identificador del tipo y el rango de definición de la variable, la lista de etiquetas lingüísticas definidas en el tipo y una representación gráfica de las funciones de pertenencia asociadas a las etiquetas (Fig. 6). Los parámetros que definen estas funciones pueden ser editados igualmente en otra ventana de la herramienta.

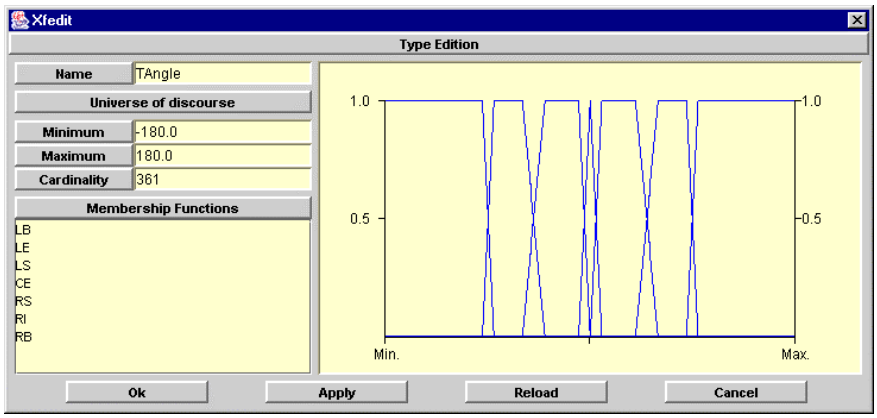

Figura 6. Ventana de edición de tipos.

La descripción de una base de reglas se realiza por medio de una ventana de edición que permite introducir el identificador de la base de reglas, el conjunto de operadores utilizado y la lista de variables de entrada y salida de la base (Fig. 7). El contenido de las reglas de la base puede ser editado de tres formas: libre, tabular y matricial. El formato libre permite definir reglas de cualquier nivel de complejidad. Por su parte, el formato tabular está diseñado específicamente para editar reglas del tipo «si x0 es X0 y x1 es X1 y .. y xn es XN entonces z es Z», es decir, formadas por igualdades y conjunciones. Por último, el formato matricial se ha diseñado para facilitar la edición de bases de reglas de dos entradas y una salida, y permite definir reglas del tipo «si a es A y b es B entonces c es C».

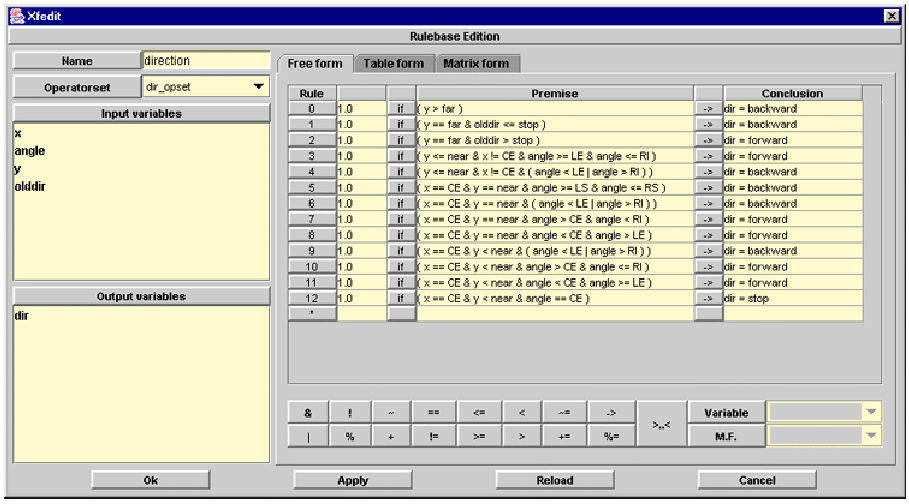

Figura 7. Ventana de edición de reglas.

La segunda de las herramientas de edición incluidas en Xfuzzy es *xfpkg*. Esta herramienta tiene como objetivo la edición de los paquetes de funciones que pueden ser utilizadas en la definición de los sistemas. Estas funciones son de cuatro tipos: funciones binarias, que pueden ser asignadas a los operadores de conjunción, disyunción, implicación y agregación de reglas; funciones unarias, relacionadas con los modificadores lingüísticos que ofrece el entorno; funciones de pertenencia, asociadas a las etiquetas lingüísticas de las variables del sistema; y métodos de concreción, que permiten obtener un valor discreto representativo para el resultado difuso de una base de reglas. La ventana principal de *xfpkg* contiene diversos paneles que permiten definir los parámetros de las funciones, sus restricciones y la definición de cada función en varios lenguajes de programación.

6. ETAPA DE AJUSTE AUTOMÁTICO

Ajustar manualmente el comportamiento de un sistema difuso resulta una tarea muy complicada debido al gran número de parámetros que determinan su comportamiento. Este problema, que no es exclusivo de los sistemas difusos, ha provocado que por parte de los investigadores se haya dedicado un gran esfuerzo al desarrollo de técnicas de ajuste automático de sistemas.

El entorno Xfuzzy 3.0 posee una herramienta, llamada *xfsl*, dedicada al ajuste automático por medio de algoritmos de aprendizaje supervisado [9]. El objetivo de estos algoritmos es ajustar el comportamiento del sistema a un comportamiento conocido, expresado por un conjunto de patrones de entrada/salida. A partir de estos patrones se genera una función de error que expresa la desviación entre el comportamiento del sistema y el descrito por los patrones. Los algoritmos de aprendizaje supervisado pretenden minimizar esta función de error.

La herramienta *xfsl* incluye un amplio conjunto de algoritmos de aprendizaje, entre los que se encuentran algoritmos de descenso por gradiente, algoritmos de gradiente conjugado, algoritmos de segundo orden, algoritmos de descenso sin derivadas y algoritmos estadísticos. La herramienta permite seleccionar, además, que parámetros del sistema se deben ajustar. *Xfsl* permite elegir entre varias funciones de error aplicables tanto a sistemas de salida continua, como los controladores difusos, como a sistemas de salida discreta, como los clasificadores difusos. Otra característica importante de *xfsl* es que incluye dos mecanismos para simplificar los resultados del ajuste. El primero consiste en eliminar reglas y funciones de pertenencia que hayan quedado obsoletas. El segundo consiste en agrupar funciones de pertenencia que hayan quedado prácticamente iguales tras el ajuste. La figura 8 muestra un ejemplo de

aprendizaje y agrupamiento de funciones de pertenencia. En la configuración inicial se parte de una superficie plana y tantas etiquetas lingüísticas en el consecuente como reglas se hayan definido. Tras el aprendizaje la superficie se ajusta a la función buscada y las funciones de pertenencia del consecuente muestran sus parámetros adecuados al problema. A continuación se reduce el número de funciones de pertenencia agrupándolas. Ello permite optimizar la base de reglas y, por lo tanto, el sistema difuso final.

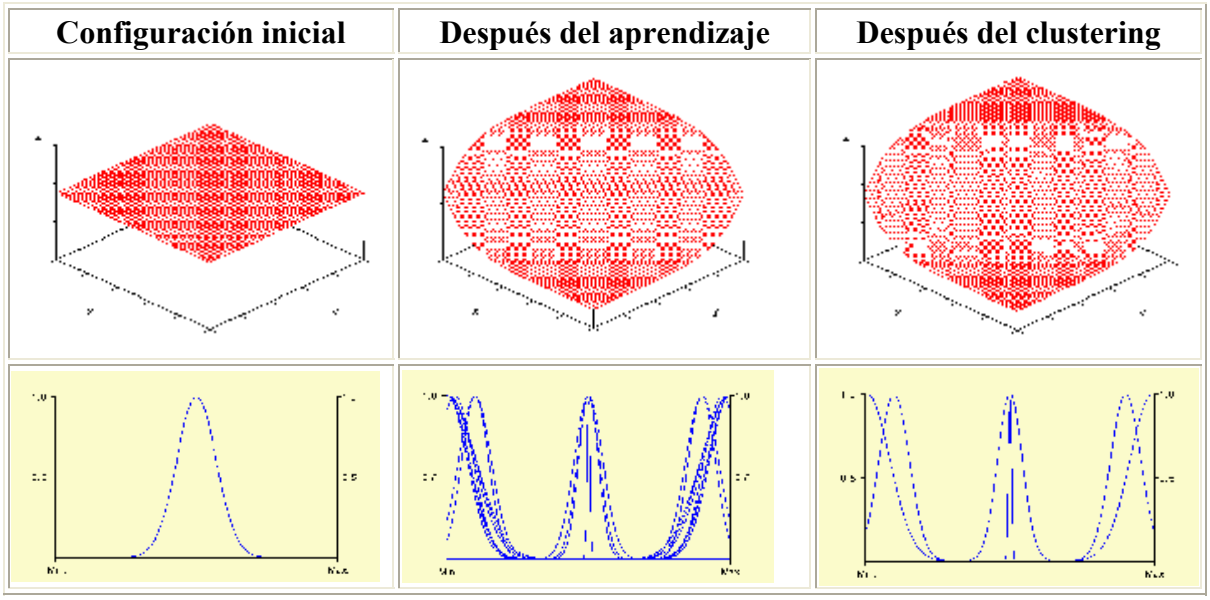

Figura 8. Ejemplo de aprendizaje y *clustering*.

En un futuro próximo se pretende que el entorno Xfuzzy cuente con nuevas herramientas que aborden otros aspectos relacionados con el aprendizaje automático, como la extracción de reglas a partir de un conjunto de patrones o la simplificación automática.

7. ETAPA DE VERIFICACIÓN

La etapa de verificación consiste en el estudio del comportamiento del sistema en desarrollo para detectar posibles errores y localizar las causas de estos fallos. El entorno Xfuzzy 3.0 contiene cuatro herramientas dedicadas a mostrar el comportamiento del sistema de diferentes formas.

La herramienta *xf2dplot* desarrolla representaciones gráficas del comportamiento del sistema en dos dimensiones. Esta herramienta permite estudiar la dependencia de una variable de salida respecto a una de las variables de entrada del sistema. La ventana principal permite seleccionar la variable de entrada y de salida a representar. Las variables de entrada no utilizadas deben ser fijadas a un cierto valor.

La segunda de las herramientas incluidas en la etapa de verificación es *xf3dplot*, que realiza representaciones gráficas en tres dimensiones (Fig.9). La ventana principal permite seleccionar las dos variables de entrada y la variable de salida que van a ser representadas, y asignar un valor a las variables de entrada no seleccionadas.

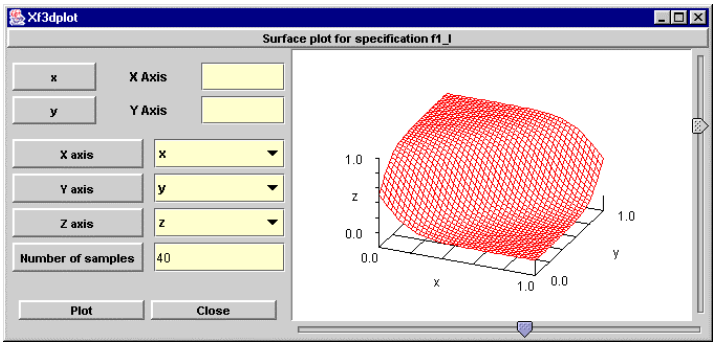

Figura 9. Ventana de *xf3dplot*.

Un aspecto importante de la verificación de un sistema difuso consiste en poder estudiar los detalles internos del proceso de inferencia para unos valores de entrada dados. Esta es la finalidad de la herramienta *xfmt*, que permite realizar una monitorización del proceso de inferencia del sistema difuso en desarrollo. La ventana principal de esta herramienta permite introducir los valores de las variables de entrada y muestra la estructura de razonamiento del sistema y los valores de salida generados.

A partir de la representación de la estructura del sistema se pueden visualizar las ventanas de monitorización de cada base de reglas (Fig. 10). En estas ventanas se representan el valor y las funciones de pertenencia de cada variable de entrada, el grado de activación de cada regla y el conjunto difuso asignado a cada variable de salida de la base de reglas. Esta herramienta permite detectar el origen de posibles errores del sistema, al poner de manifiesto qué reglas se activan en cada momento y qué valores toman cada una de las variables del sistema en el proceso de inferencia.

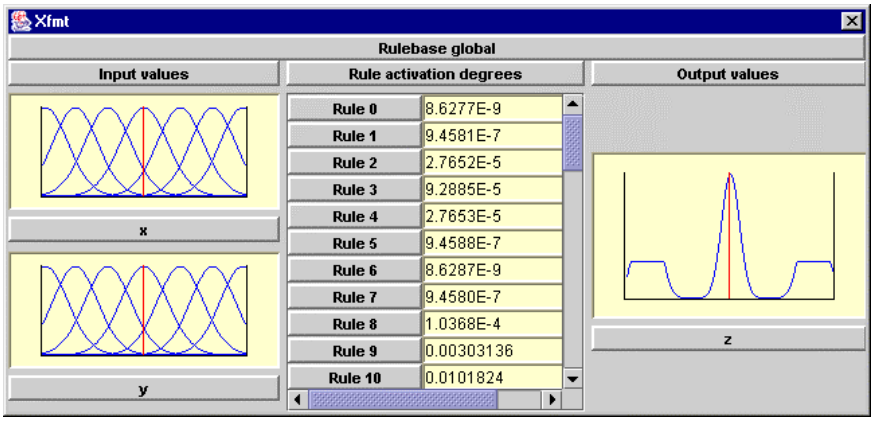

Figura 10. Monitorización de una base de reglas.

La etapa de verificación se completa con una herramienta de simulación, llamada *xfsim*, que muestra el comportamiento del sistema difuso cuando se conecta a un sistema externo. Este sistema puede ser un modelo del sistema real en el que se desea implantar el sistema difuso tras la etapa de síntesis, o incluso el sistema real si éste se conecta al equipo de diseño mediante algún sistema de adquisición de datos. La herramienta necesita que el sistema externo sea representado mediante una clase Java que desarrolle una determinada interfaz. La creación y compilación de esta clase es responsabilidad del usuario. La ventana principal de la herramienta permite seleccionar la clase Java que representa el sistema externo, los valores iniciales de este sistema y los límites de la simulación (Fig. 11).

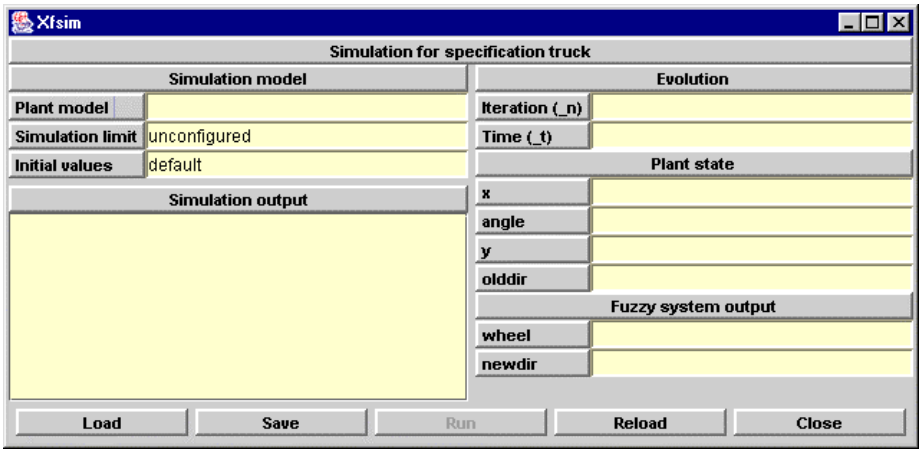

Figura 11. Ventana de *xfsim*.

Xfsim permite obtener dos tipos de salida: ficheros de datos y representaciones gráficas. La configuración de la herramienta permite generar ficheros de datos donde se almacenen las variables de simulación seleccionadas por el usuario. Las representaciones gráficas permiten mostrar la evolución de cualquiera de estas variables o la relación entre dos de estas variables de simulación (Fig.12).

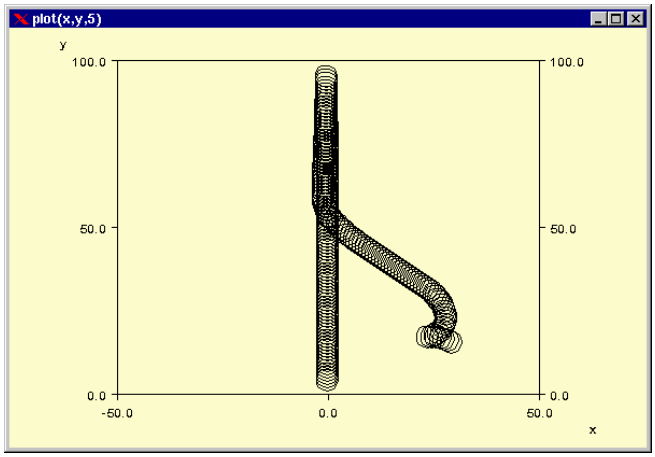

Figura 12. Un ejemplo de representación gráfica de *xfsim* mostrando la evolución del sistema.

8. ETAPA DE SÍNTESIS

La etapa final del proceso de diseño de un sistema difuso consiste en generar una realización del sistema que pueda ser implantada en el entorno de funcionamiento para el que ha sido diseñado. Esta realización puede ser de dos tipos: software y hardware. En las realizaciones software el objetivo es generar una representación del sistema difuso en un lenguaje de programación de alto nivel. Esta solución es muy flexible, ya que no impone restricciones a la complejidad del sistema diseñado y puede ser incorporada como módulo difuso en un proyecto software de mayor alcance. Por su parte, la realización hardware consiste en producir circuitos microelectrónicos que realicen el proceso de inferencia descrito por el sistema difuso. Esta opción presenta una velocidad de inferencia más alta y un consumo de área y potencia mucho más bajo que la realización software.

El entorno Xfuzzy 3.0 ofrece tres herramientas dedicadas a la etapa de síntesis, llamadas *xfc*, *xfcpp* y *xfj*. Estas tres herramientas están dirigidas hacia la síntesis software y generan representaciones del sistema difuso en los lenguajes C, C++ y Java, respectivamente. En la actualidad se está desarrollando herramientas de síntesis hardware, con una funcionalidad

similar a la herramienta incorporada en la versión anterior del entorno [10]. La nueva versión Xfuzzy 3.1b añade además nuevos esquemas de síntesis de circuitos permitiendo obtener soluciones en problemas complejos que requieren aplicación de metodologías de codiseño hardware&software. La figura 13 muestra un diagrama del flujo de diseño de sistemas difusos sobre FPGA. La herramienta de síntesis *xfvhdl* genera la descripción VHDL a partir de una especificación XFL. A continuación dicha descripción VHDL puede ser sintetizada y simulada mediante herramientas estándar de diseño de circuitos.

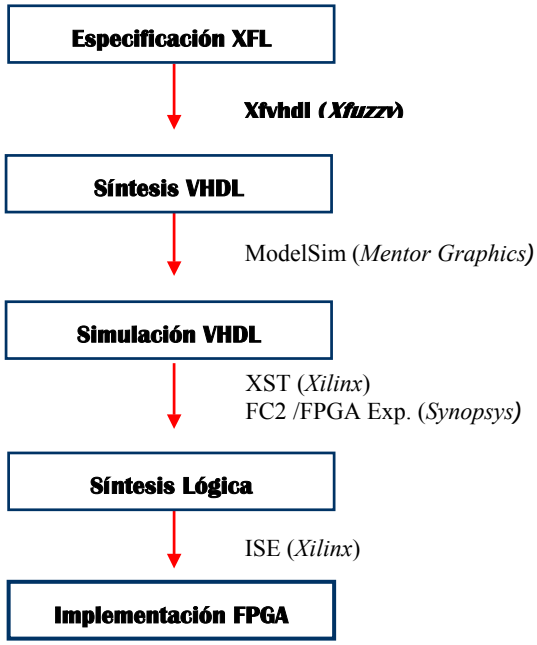

Figura 13. Diagrama del flujo de diseño con Xfuzzy.

9. CONCLUSIÓN

El entorno Xfuzzy 3.0 proporciona un conjunto de herramientas que cubren en su mayor parte las diferentes etapas y aspectos relacionados con el diseño de sistemas de inferencia difusa. Estas herramientas en su conjunto desarrollan un entorno flexible y potente que ofrece una interfaz gráfica homogénea que permite diseñar sistemas difusos complejos por medio de un lenguaje de descripción formal que admite bases de reglas jerárquicas y modificadores lingüísticos, conectivas difusas, funciones de pertenencia y métodos de concreción extensibles por el usuario. El entorno se distribuye libremente bajo licencia pública GNU y está totalmente programado en Java, por lo que puede ser ejecutado en la mayoría de sistemas informáticos.

Las características del entorno Xfuzzy 3.0 hacen que sea una herramienta muy adecuada como apoyo a la docencia de materias relacionadas con la lógica difusa. El entorno es abierto en el sentido que permite desarrollar fácilmente aplicaciones que requieran ejecutar un motor de inferencia difuso, así como modificar y añadir nuevos formalismos matemáticos para la especificación del sistema.

10. BIBLIOGRAFÍA

[1] Baturone, I., Barriga, A., Sánchez-Solano, S., Jiménez, C. J. López, D. R., "Microelectronic Design of Fuzzy Logic-Based Systems", CRC Press, 2000.

[2] D.R. López, C.J. Jiménez, I. Baturone, A. Barriga, S. Sánchez Solano, "Xfuzzy: A Design Environment for Fuzzy Systems", Proc. 7th IEEE Int. Conf. on Fuzzy Systems (FUZZIEEE'98), pp. 1060-1065, Anchorage, 1998.

[3] F.J. Moreno, I. Baturone, S. Sánchez-Solano, A. Barriga, R. Senhadji, "El entorno Xfuzzy 3.0 de diseño de sistemas difusos", XI Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'2002), León, Sept. 2002.

[4] D.R. López, F.J. Moreno Velo, A. Barriga, S. Sánchez Solano, "XFL: A Language for the Definition of Fuzzy Systems", Proc. 6th IEEE Int. Conf. on Fuzzy systems (FUZZIEEE'97), pp. 1585-1591, Barcelona, 1997.

[5] S. Sánchez Solano, A. Barriga, I. Baturone, D. R. López, F. J. Moreno Velo, "Curso de Diseño de Controladores Difusos Mediante el Entorno de Desarrollo Xfuzzy", IV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE, 2000), Barcelona, 13-15 Sept. 2000.

[6] A. Barriga, S. Sánchez Solano, I. Baturone, A. Bermúdez, "Libro electrónico para el autoaprendizaje de la lógica difusa", V Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE, 2001), pp. 29-32, Las Palmas, Feb. 2002.

[7] Xfuzzy home page: http://www.imse.cnm.es/Xfuzzy

[8] F.J. Moreno Velo, S. Sánchez Solano, A. Barriga, I. Baturone, D.R. López, "XFL3: An Specification Language for Fuzzy Systems", Mathware & Soft Computing, Vol. VIII, n. 3, pp. 239-253, 2001.

[9] F.J. Moreno Velo, I. Baturone, S. Sánchez Solano, A. Barriga, "XFSL: A Tool for Supervised Learning of Fuzzy Systems", Proc. European Symposium on Intelligent Technologies, Hybrid Systems an their implementation on Smart Adaptive Systems (EUNITE-2001), pp. 58, Tenerife, 2001.

[10] E. Lago, C.J. Jiménez, D.R. López, S. Sánchez Solano, A. Barriga, "Xfvhdl: A Tool for the Synthesis of Fuzzy Logic Controllers", Proc. Design, Automation and Test in Europe (DATE'98), pp. 102-107, París, 1998.