COMPONENTE DE SISTEMA DE INFERENCIA DIFUSA (FIS) PARA EVOLUCIÓN 3.5

GESMAN DAVID MACHADO MENDOZA.
CESAR EDUARDO GONZÁLEZ PÉREZ.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2006

COMPONENTE DE SISTEMA DE INFERENCIA DIFUSA (FIS) PARA EVOLUCIÓN 3.5.

GESMAN DAVID MACHADO MENDOZA.
CESAR EDUARDO GONZÁLEZ PÉREZ.

Trabajo de grado para optar por el título de:
Ingenieros de Sistemas

Director
HUGO HERNANDO ANDRADE SOSA
Ingeniero de Sistemas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2006

DEDICATORIA

Mis más sinceros agradecimientos a:
Mi madre Zoraida ("El General"), mi padre Dissan ("Bruno") por el apoyo que siempre le han brindado a los sueños de sus hijos, anteponiéndolos a los suyos. Al profesor Hugo Andrade por brindarnos su confianza. A mis hermanos, ñoño y Alain, por estar siempre dispuestos alentarme en los momentos difíciles. Por ultimo a todos mis amigos, con los cuales formé una familia y un muy buen equipo de microfutbol (los YUK's) en la ciudad de Bucaramanga.

Gesman Machado Mendoza.

DEDICATORIA

A mis padres por ser el pilar que ha sostenido en mí la idea de que siempre es posible, a mis hermanas que son sin duda la prueba de que Dios existe, a mi inmensa familia que incondicionalmente ha creído y ha brindado su apoyo cuando le he necesitado, familia que crece con esas personas que de vez en cuando me llaman amigo, esas que llevan de apellido los YUK´s. A esas personas les puedo decir que las llevo en mi corazón. Agradezco al profe Hugo por confiar en nosotros y al grupo SIMON por su apoyo.

Cesar Eduardo González Pérez

)O

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO
COMPONENTE DE SISTEMA DE INFERENCIA DIFUSA (FIS) PARA EVOLUCION 3.5*

AUTORES
Gesman David Machado Mendoza.
Cesar Eduardo González Pérez**

PALABRAS CLAVES
Sistema de Inferencia Difusa, Lógica Difusa, Simulación, Dinámica de Sistemas, Software.

DESCRIPCIÓN

La Dinámica de Sistemas orienta el proceso de construcción de un modelo matemático estructural de un fenómeno, y posibilita simular su comportamiento dinámico en el transcurrir del tiempo, o de otra variable independiente. Para apoyar este proceso el grupo SIMON de investigaciones de la Universidad Industrial de Santander desarrollo EVOLUCION, software que permite modelar y simular con Dinámica de Sistemas.

En este documento se presentan en detalle las actividades desarrolladas en el proceso de construcción e integración a EVOLUCION, de un modulo software, que permita el trabajo con Sistemas de Inferencia basados en Lógica Borrosa (FIS por sus siglas en Ingles), como si tratase de un elemento mas del diagrama Forrester.

Dicho software esta dotado de dos componentes, un editor que permite al usuario la creación y puesta a punto de Sistemas de Inferencia basados en Lógica Borrosa tipo Mandani, además de un componente de nombre FIS, el cual es el encargado de realizar los cálculos pertinentes a este elemento. Por su diseño, el modulo puede ser usado en el desarrollo de otras aplicaciones que requieren implementar Sistemas de Inferencia basados en Lógica Borrosa tipo Mandani.

---

* Trabajo de grado
** Facultad de Ingenierías Físico Mecánicas, Escuela de Ingeniería de sistemas e informática. Ing. Hugo Hernando Andrae Sosa.

ABSTRACT

TITLE
COMPONENT OF FUZZY INFERENCE SYSTEM (FIS) FOR EVOLUCION 3.5.

AUTHORS

Gesman David Machado Mendoza.
Cesar Eduardo González Pérez.

KEY WORDS

Fuzzy Interference System, Fuzzy logic, Dynamics of Systems, simulation, Software.

DESCRIPTION

The Dynamics of Systems guides the construction process of a structural mathematical model of a phenomenon, and it facilitates to simulate their dynamic behaviours in lapsing of the time, or another independent variable. For supporting this process the SIMON research group developed EVOLUCION, a Software which allow the modelling and simulation of dynamics of systems.

In this document are presented detailed the activities developed in the construction process and integration to EVOLUCION, from a software module, that allow working with Fuzzy Interference System (FIS) based in Fuzzy logic, like and additional element from a Forrester Diagram.

This software is attached with two components. The foremost of them is an editor which allow the user create and set about to based Systems of Inference in Fuzzy Logic type MANDANI, furthermore of a FIS named component, which is in charge of carrying out the pertinent calculations to this element. It design allows be used in development of another applications that requires do Systems of Inference in Fuzzy Logic type MANDANI.

---

* Undergraduated thesis.
** Faculty of Physiomechanical Engineerings , School of Engineering of systems and computer science. Eng Hugo Hernando Andrae Sosa.

INTRODUCCIÓN

Desde hace 4 años el grupo SIMON de investigaciones de la Universidad Industrial de Santander, viene estudiando la forma de relacionar la Dinámica de sistemas (D.S.) y lo que el grupo ha llamado otras herramientas matemáticas para la representación del conocimiento. En este proceso investigativo el componente FIS, representa la integración de la Lógica difusa y el modelo con Dinámica de Sistemas con EVOLUCION. Con el fin de dar a conocer a la comunidad internacional que trabaja con Dinámica de Sistemas, los resultados de este proyecto, se encuentra en fase de aprobación una ponencia en el IV Congreso Latinoamericano de Dinámica de Sistemas en Cancún México.

El componente FIS, dota a EVOLUCION de una herramienta para la creación de Sistemas de Inferencias Difusa, sin necesidad de tener que recurrir a ningún otro tipo de ayuda software o labores de programación. Todo esto con el fin de dotar a los modeladores de D.S. de herramientas que le permitan enfrentarse a la complejidad de la realidad con mejores posibilidades para su representación.

Estas memorias están organizadas en 8 capítulos los cuales sintetizan las actividades y experiencias vividas en el desarrollo del proyecto.

ASPECTOS GENERALES. Se presentan los antecedentes del proyecto, la metodología usada para el desarrollo del mismo y se describe en términos generales el componente FIS.

MARCO TEÓRICO. Se describe el enfoque de desarrollo de software utilizado para la realización del componente FIS. Se resumen los conceptos

básicos de la lógica difusa, relacionados con la construcción de FIS. En este tema se hace un poco de énfasis, por no ser una información no común en la comunidad de D.S. y para contribuir a la difusión del aporte de la integración de la D.S con la lógica difusa.

FASE DE FORMACIÓN. En este capítulo se presenta una síntesis de las actividades y los objetivos que se lograron en esta primera etapa del proyecto.

FASE DE DESARROLLO. En este capitulo se presenta la documentación del proceso de desarrollo de software, seguido para la construcción del componente FIS.

FASE DE PRUEBAS. Se presentan en detalle los objetivos y resultados de las pruebas del FIS.

CONCLUSIONES. Reflexiones después de haber terminado este proyecto.

RECOMENDACIONES. Para el uso del los productos del proyecto y para dar luces de hacia donde deben apuntar los futuros desarrollos del componente FIS.

BIBLIOGRAFÍA. Contiene las referencias bibliográficas que se utilizaron durante el proceso del proyecto.

1. ASPECTOS GENERALES

1.1 ANTECEDENTES Y DESCRIPCIÓN DEL PROBLEMA

Analizar situaciones del mundo para emitir juicios que son en definitiva, alternativas de solución a problemas que se han presentado, que se presentan o que se espera que se presenten, es el fin ultimo del trabajo de los profesionales en ciencias aplicadas; juicios que cambian favorable o desfavorablemente, en pequeña o gran medida ese mundo. El hecho de utilizar la palabra *juicio*, es suficiente para expresar la subjetividad inherente al proceso de análisis (construido por una persona o por un colectivo de personas), es decir, que hacen parte del proceso los intereses personales (situación moral de la persona o del grupo de personas) y la capacidad de los sentidos o instrumentos de percepción y del motor[1] que interpreta o resuelve (situación de formación científica). La complejidad de las situaciones y las limitaciones de los sentidos hace indispensable el apoyo en herramientas que permitan el mejor entendimiento de aquellas, herramientas lógicas como, números, gráficos, etc., y herramientas físicas como, papel, computadores, etc., o una combinación de ambas, que trascienden y se convierten en lenguajes específicos para cada tipo de situación.

Es así como aparece el modelado con dinámica de sistemas, en afán de estudiar situaciones complejas[2] y como alternativa a otras metodologías de modelado, luego en la UIS aparece EVOLUCION, una herramienta software para crear modelos con aquella filosofía. Con el mismo objetivo se sigue enriqueciendo la herramienta; necesidades que surgen en la medida en que

[1] Modelos mentales individuales y colectivos
[2] situaciones con múltiples elementos, relaciones entre elementos, y estados, que se vuelven difíciles de solucionar con métodos convencionales.

la complejidad de los modelos crece[3], guían el proceso; ejemplos de ellos son los modelos de sistemas sociales, los cuales llevan consigo variables que son fácilmente evaluadas por los sentidos, pero que son difíciles de expresar en un modelo de simulación por computador debido a la dificultad para cuantificarlas, variables tan comunes como, el estado de ánimo de una persona o la calidad académica de los estudiantes UIS, etc.; normalmente se hace referencia a ellas como variables cualitativas.

Para tratar variables cualitativas se buscó apoyo en las herramientas matemáticas existentes y se encontró que la lógica de conjuntos difusos presenta una forma para representarlas y además ofrece un conjunto de operaciones para dichas variables; EVOLUCION permite la integración de esta nueva herramienta matemática, a través de multiplicadores que representan comportamientos no lineales y variables exógenas, el modelo con estas características fue llamado por investigadores de grupo SIMON como *modelo apoyado*. Como alternativa a esta forma de integración se utilizaron librerías dinámicas (DLLs), las cuales llevan implementadas funciones que emulan el comportamiento de un sistema de inferencia difusa (FIS, por su nombre en ingles) para un problema en particular, el modelo con estas características fue llamado *modelo integrado*. El modelo apoyado tiene la desventaja que el comportamiento de los valores incluidos no es dinámico, lo que hace que el modelo no se acerque tanto como se desea a la situación observada, esta falencia es superada por el modelo integrado; pero ambos tienen una desventaja común, se debe recurrir a otras herramientas software para encontrar las funciones o valores correspondientes a esas utilidades.

Además, crear un modelo integrado demanda un conocimiento en programación de computadores, ya que es necesario tomar los valores

[3] Pensamiento sistémico: Diversidad en búsqueda de Unidad – Pág. 189: Modelado Dinámico-Sistémico.

generados por herramientas que permitan el modelado con Lógica Difusa, como MATLAB, UNFUZZY, FUZZYTECH, etc.; estructurarlos y luego edificar una DLL que pueda ser utilizada por EVOLUCION. Este proceso es posible sólo si se es experto en programación de computadores o si se tiene un conocimiento similar; en general las personas que comúnmente están involucradas en el modelado con Dinámica de Sistemas y Lógica Difusa no son programadores expertos, lo que implica que estas adquieran esos conocimientos o recurran a personas con el perfil indicado para realizar todo el proceso, haciendo que los modelos requieran más tiempo de lo necesario para su construcción y agregándole una nueva variable al proceso, esta última puede expresarse así: ¿en qué medida la dll reproduce el comportamiento de la herramienta que se tomó como referencia para crearla?

Por lo anterior se plantea agregar a EVOLUCION las herramientas necesarias para crear modelos bajo las filosofías de modelado integradas, Dinámica de sistemas y Lógica Difusa.

1.2 OBJETIVOS DEL PROYECTO.

1.2.1 Objetivo General. Desarrollar e integrar a EVOLUCIÓN un componente software, que permita la implementación de sistemas de inferencia difusa (FIS) para la definición de las variables de los modelos de dinámica de sistemas.

1.2.2 Objetivos Específicos. Diseñar un componente software expresado a través de UML, que permita la elaboración de Sistemas de Inferencia Difusa, tal que:

- El diseño sea orientado a objetos y basado en componentes siguiendo el enfoque y la documentación contemplado en EVOLUCION 3.5.

- El componente pueda ser utilizado en otras aplicaciones que lo requieran.

- Las variables de membresía puedan ser descritas gráficamente[4].

- Los resultados puedan ser representados gráficamente[5].

- Desarrollar el componente software para la creación de Sistemas de Inferencia Difusa diseñado.

- Integrar a EVOLUCIÓN el componente software para la creación de Sistemas de Inferencia Difusa, que se desarrolle, para la creación dinámica e integrada de modelos con Dinámica de Sistemas y Lógica Difusa, considerando que:

- EVOLUCIÓN debe conservar sus características y funcionalidades.

- Documentar el componente software, así como la interacción con EVOLUCION.

- Evaluar el software tomando como referencia pruebas con usuarios novatos y experimentados para detectar posibles errores y/o dificultades en el uso del componente.

[4] La representación gráfica de una función de membresía es en dos dimensiones, donde la variable independiente corresponde al universo de discurso, su rango de valores depende de las especificaciones del modelador; la variable dependiente es el grado de pertenencia de la variable al conjunto, sus posibles valores están en [0 1].
[5] La representación gráfica de las resultados es en dos dimensiones, donde se representa la dependencia entre una función de salida y una de las entradas

Evaluar el componente efectuando pruebas que midan su efectividad, comparándolas con UNFUZZY y MATLAB, con el fin de corroborar la confiabilidad de sus resultados.

1.3 JUSTIFICACIÓN Y VIABILIDAD DEL PROYECTO

1.3.1 Justificación. Se justifica el desarrollo del proyecto desde dos perspectivas, una que hace referencia a la forma como EVOLUCION 3.5 permite la integración de Dinámica de Sistemas y Lógica Difusa, y la otra hace referencia al proyecto de Ingeniería del Software inherente al proceso de integración de un nuevo componente a EVOLUCION 3.5.

Cabe resaltar que no existe aún conocimiento de herramientas software que permitan la creación de modelos integrados de dinámica de sistemas y lógica difusa. Lo que convierte este trabajo en un proyecto de innovación.

Dinámica de sistemas-lógica difusa con evolución 3.5. En un modelo desarrollado con EVOLUCION 3.5 es posible hacer uso de los Sistemas de inferencia difusa (FIS) de dos maneras: una construyendo una forma tabular que se genera con el FIS (en otra herramienta software) y que contiene todos los posibles valores de entrada y salida del FIS. La otra forma es agregando al software el FIS en término de una función, mediante una DLL.

Estas formas de integración entre la dinámica de sistemas y la lógica difusa tienen inconvenientes y limitaciones, algunas de estas se enuncian a continuación:

Limitaciones en el número de entradas de las formas tabulares, se refiere a que estas sólo permiten manejar una entrada lo que elimina la posibilidad de utilizar una cantidad incalculable de FIS, dado que este tipo de sistemas son

capaces de funcionar con n entradas, forzando al experto a compactar o relacionar de manera antinatural los diferentes elementos de entrada que necesita el sistema de inferencia para operar.

Otra dificultad que se presenta es la obligatoriedad de un igual espaciado entre los valores de entrada para ajustarse al mecanismo del punto inicial y el paso para representar la no-linealidad, característica deseable en caso que un sector del FIS requiera mayor detalle que otro.

Adicionalmente, las tablas presentan otro inconveniente en la utilización de FIS, el número de sus salidas se encuentra siempre restringido a uno, situación que obliga al modelador a crear múltiples tablas alimentadas por la misma entrada y que operarían independientemente, complicando de esta manera el proceso de construcción del modelo y generando redundancia de trabajo al emplear múltiples elementos para representar un solo FIS.

Otra forma de incluir un FIS en EVOLUCION es aprovechando la posibilidad que brinda este para incluir DLL's (definiendo una función que puede ser llamada desde cualquier modelo para definir una o más de sus variables). Esto genera algunos problemas:

- Muchas veces obliga a que en el equipo de trabajo que realiza el modelo deba haber una persona que tenga conocimiento de programación de computadores para que desarrolle las DLL's que implementan el FIS.

- Para realizar un cambio en el FIS por pequeño que este sea; implica generar una DLL nueva lo cual genera perdida de tiempo y en algunos casos gastos económicos al tener que pagarle a un desarrollador para que realice la nueva DLL.

Ingeniería Del Software. Para lograr el objetivo del proyecto se hace necesario tener un fundamento matemático, que permita el entendimiento de la complejidad asociada a la teoría de conjuntos borrosos, así como la creación de modelos basados en esta teoría. Se debe también tener fundamento en programación de computadores, para dar solución a situaciones que requieran de análisis numérico apoyado en herramientas computacionales (situaciones comunes en desarrollos de software de simulación).

Agregar un nuevo componente a EVOLUCION 3.5 requiere de un conocimiento en Ingeniería del Software que permita el entendimiento[6] del software actual y la creación de alternativas de solución al problema, expresadas a través de modelos en el Lenguaje Unificado de Modelado[7].

Todas estas características son fortalezas del Ingeniero de Sistemas – UIS y corresponden con su perfil.

1.3.2 Viabilidad. La disponibilidad de recursos humanos y tecnológicos, la facilidad de encontrar información relevante y, la formación académica y humana de los desarrolladores que los hace idóneos para afrontar la complejidad del tema, hace viable el desarrollo el proyecto.

La Universidad Industrial de Santander a través del grupo SIMON ofrece su apoyo al desarrollo del proyecto, expresado en la riqueza de su recurso humano, en sus instalaciones físicas que facilitan la labor investigativa, en la

[6] Entendimiento de diagramas de componentes, de clases, de objetos, de estados, etc.
[7] Se hace referencia a UML dado que ha establecido como estándar para representar el proceso de desarrollo de software orientado a objetos. Pero destacando que el Ingeniero de Sistemas UIS tiene la capacidad de aprender rápidamente y adaptarse a los cambios que el entorno genera.

bibliografía existente y destacando que la dinámica de trabajo del grupo, hace que exista apoyo constante entre sus integrantes.

El grupo SIMON posee toda la información generada a partir del desarrollo de EVOLUCION 3.5, también cuenta con el apoyo de sus desarrolladores, quienes pueden enriquecer el proceso con su experiencia; además dispone de material bibliográfico útil para el desarrollo del proyecto, tal como, el Proyecto de grado: MODELO CONCEPTUAL GENERALIZADO DE UN SISTEMA PARA SOPORTAR LA TOMA DE DECISIONES, Enfoque integrador entre la Dinámica de Sistemas y la Inteligencia Artificial por Guillermo Luque y Vladimir Pradilla, 2003 y el proyecto, APLICACIONES DE UN "MODELO CONCEPTUAL GENERALIZADO DE UN SISTEMA PARA SOPORTAR LA TOMA DE DECISIONES", Enfoque Integrador Entre la Dinámica de Sistemas y la Inteligencia Artificial por César Augusto Rivera Palacios, Freddy Sarmiento Villamizar.

La Universidad Industrial de Santander a través de la escuela de Ingeniería Química cuenta con la herramienta MATLAB que sirve de apoyo a las pruebas del componente FIS; existe también UNFUZZY, una herramienta para el modelado con sistemas de inferencia difusa desarrollada en la Universidad Nacional de Colombia y que puede utilizarse como apoyo a las pruebas del componente FIS.

La biblioteca central de la UIS cuenta con un número importante de libros, revistas y tesis con temas como, Ingeniería y desarrollo de software.

En lo relacionado con los desarrolladores se puede decir que:

El fundamento matemático de los desarrolladores les permite entender la teoría relacionada con la lógica de conjuntos difusos, para luego junto con el

fundamento en modelado de sistemas dinámicos, modelar situaciones que aporten al proceso; parte trascendental para el desarrollo del proyecto.
Los fundamentos en ingeniería y desarrollo de software hacen posible el desarrollo del proyecto.

1.4 COMPONENTE FIS. COMPONENTE, SISTEMA DE INFERENCIA DIFUSA

El componente FIS[8], es un importante paso en el camino de integrar la D.S. con otras matemáticas[9]. En este proceso de integración, el papel que juega el componente FIS, es el de integrar la Lógica difusa a la D.S.

El componente FIS, permite la creación de sistemas de inferencia borrosa tipo Mandami[10], con todo lo que esto implica, crear variables lingüísticas de entrada y salida, definir los operadores INTERSECCIÓN, UNIÓN, IMPLICACIÓN, AGREGACIÓN, COMPLEMENTO y método de desborrosificación; dentro del ambiente de trabajo de evolución 3.5, eliminando por completo el uso de otras herramientas como Matlab y UNFUZZY, usadas hasta el presente en SIMON para desarrollar los ejercicios de integración. Dándole así al modelador mayor facilidad para probar diferentes configuraciones de FIS para el mismo modelo, librándolo de los inconvenientes descritos en el apartado 1.3.1.1.

1.5 METODOLOGÍA

Se plantean dos fases, una de formación y otra de Software así:

[8] FIS Sistema de Inferencia Difusa.
[9] Lógica difusa, Inteligencia Artificial, Algoritmos Genéticos, Redes Neuronales.
[10] Un sistema de inferencia tipo Mamdani se corresponde con la noción más conocida de *sistemas borrosos;* está compuesto por una base de conocimiento, un motor de inferencias, y unas interfaces de borrosificación y desborrosificación. Tipo de sistema de inferencia difusa que utiliza

1.5.1 Primera fase: formación. En esta etapa se tienen como objetivos:

- Conocer a fondo el fundamento teórico de la lógica borrosa y su utilización en el estudio de situaciones reales.

- Estudio del Lenguaje Unificado de Modelado (UML) para lograr que el diseño y la documentación del proyecto estén en un lenguaje estándar y facilitar la tarea de dar continuidad al proyecto en el futuro.

- Aprender el lenguaje y/o entorno de programación que se va a utilizar para el desarrollo del proyecto.

- Revisión de la bibliografía para buscar antecedentes de proyectos de este tipo.

1.5.2 Segunda fase: proceso de software. Teniendo en cuenta que lo que se quiere es una nueva versión de EVOLUCION, se hace indispensable conocerlo a fondo para identificar los aspectos más importantes que puedan servir de apoyo al desarrollo del proyecto.

Para desarrollar el componente software, se eligió la metodología espiral dado que le da un tratamiento especial al riesgo existente en el ciclo de vida del proyecto y porque se requiere un prototipo en funcionamiento para el desarrollo de trabajos en el grupo SIMON, así como para el apoyo de la materia sistemas dinámicos en la escuela de Ingeniería de sistemas.

La metodología espiral presenta seis grupos de actividades que guían el proceso, que son:

- ✓ Comunicación con el cliente

- ✓ Planificación
- ✓ Análisis de riesgo
- ✓ Ingeniería
- ✓ Construcción y acción.
- ✓ Evaluación del cliente

Este grupo de actividades es vigente en cada iteración, generando nuevas versiones del software.

Se entregarán tres prototipos, la descripción de las tareas para este son:

Primer prototipo:

Comunicación con el cliente[11]***.*** **P**ara el éxito del proyecto es necesario que exista comunicación constante entre los desarrolladores y los clientes, por esto se hace indispensable definir tareas que guíen el buen entendimiento de, necesidades, sugerencias, críticas, puntos de vista y demás aspectos que en el proceso de desarrollo del proyecto aparezcan. para esta etapa se definen las siguientes tareas**:**

- Concertar los alcances que deberá tener este prototipo. Un posible aspecto relevante que podría tomarse como referencia para lograr este objetivo es “la condición que debe tener el software para explicar brevemente el funcionamiento de un FIS”.

- Identificar posibles sugerencias para la definición de interfaces de usuario que los integrantes interesados del grupo SIMON planteen. Esta tarea se puede lograr con reuniones cortas y específicas para cada grupo de

[11] Como clientes se hace referencia a: El grupo SIMON, los estudiantes de Sistemas Dinámicos II del programa ingeniería de sistemas-UIS.

interfaces (Entradas y Salidas), cortas para que se tenga plena atención y específicas para sacar el máximo provecho al ejercicio.

- Identificar posibles sugerencias para la definición de interfaces de usuario que los estudiantes de sistemas dinámicos II puedan ofrecer. Esta tarea se debe hacer después que en la materia se haya tratado el tema de modelado con dinámica de sistemas y sistemas de inferencia difusa.

Planificación: *e*valuar la complejidad del desarrollo del prototipo, para la definición de horarios y la distribución de tareas entre los desarrolladores que permitan el cumplimiento de los plazos que se especificados en el cronograma de actividades.

Análisis de riesgo: *a*quí se tienen en cuenta dos clases de riesgos, los técnicos, que podrían ser, que el prototipo a desarrollar no cumpla con los requisitos, que el prototipo esté limitado a cierta plataforma computacional, etc.; el otro tipo de riesgos tenidos en cuenta son los de gestión, como por ejemplo, en el transcurso del desarrollo del prototipo no se alcanzaron metas propuestas que podrían hacer que no se cumpla con el plazo te entrega concertado, que alguno de las personas o grupo de personas que intervienen en el proceso queden fuera transitoria o permanentemente, etc. Para evaluar estos tipos de riesgos se plantean las siguientes actividades:

- En el eventual caso que no se cumpla con alguno de los requisitos, se deben identificar las causas del hecho, definir políticas que apunten a mejorar el proceso y por último replantear el plan de trabajo de tal manera que se llegue a la consecución de los objetivos sin salirse de los lineamientos generales planteados inicialmente.

- Si existe un cambio relevante en los recursos contemplados inicialmente y perjudicial al proyecto las actividades a seguir son, determinar si es posible llegar al estado anterior y bajo qué condiciones se haría; evaluar la situación, con el cliente preferiblemente; redefinir el plan de proyecto acorde con las nuevas condiciones.

Ingeniería:

a) Análisis:

- ✓ Identificación de los procesos más representativos del sistema que se quiere desarrollar, teniendo en cuenta los alcances del prototipo.
- ✓ Estudio del entorno inmediato del componente FIS.
- ✓ Estudio de posibles interfaces de interacción con el usuario.
- ✓ Estudio de posibles interfaces de interacción entre el que se quiere desarrollar, tratando de ver en lo posible más aya del prototipo en desarrollo.
- ✓ Informe de análisis.

b) Diseño:

- ✓ Diseño de la estructura general del componente, así como de la estructura general de sus módulos, definiendo el entorno de cada uno de estos últimos.
- ✓ Diseño de las características de interfaces de comunicación módulo-módulo, componente FIS-EVOLUCION 3.5.
- ✓ Diseño de interfaces de interacción con el usuario.
- ✓ Informe de Diseño.

c) Construcción acción:

- Implementación.

✓ Codificación del prototipo FIS
✓ Integración del Prototipo de FIS con EVOLUCION.

- Pruebas.

✓ Diseño de pruebas a realizar.
✓ Ejecución de la fase de pruebas.
✓ Evaluación de las pruebas.
✓ Análisis de resultados de las pruebas.

Evaluación con el cliente.

✓ Diseño de pruebas con los clientes.
✓ Ejecución de la fase de prueba con los clientes.
✓ Evaluación de la pruebas.
✓ Análisis de resultados de las pruebas.

Segundo prototipo.

Comunicación con el cliente:

✓ Concertar los alcances que deberá tener este prototipo.
✓ Identificar posibles sugerencias generales que los usuarios puedan ofrecer, como resultado de la evaluación del primer prototipo.

Planificación. Evaluar la complejidad del desarrollo del prototipo, para la definición de horarios y la distribución de tareas entre los desarrolladores que permitan el cumplimiento de los plazos que se especificados en el cronograma de actividades.

Análisis de riesgo. Para esta fase, se plantean las mismas tareas que las de su equivalente en el primer prototipo.

Ingeniería:

Análisis:

- ✓ Estudio de las conclusiones de la fase de pruebas con el cliente del primer prototipo
- ✓ Estudio de las características que debe tener el segundo prototipo.
- ✓ Informe de análisis.

Diseño:

- ✓ Diseño de mejoras, basadas en los resultados de las pruebas.
- ✓ Diseño de las nuevas características que debe tener el prototipo.
- ✓ Informe de Diseño.

a) Construcción y acción.

Implementación:

- ✓ Codificación de las nuevas características.
- ✓ Integración del Prototipo de FIS con EVOLUCION.

Pruebas: para esta fase, se plantean las mismas tareas que las de su equivalente en el primer prototipo.

Evaluación con el cliente: para esta fase, se plantean las mismas tareas que las de su equivalente en el primer prototipo.

Tercer prototipo: el desarrollo de este prototipo tiene la misma estructura que la del segundo prototipo.

2. MARCO TEÓRICO

2.1 INTRODUCCIÓN

Para el desarrollo de este proyecto se han tendido en cuenta aspectos técnicos correspondientes a la Ingeniería del Software, como las metodologías desarrollo y diseño que guiaron el proceso. A demás son indispensables los fundamentos de la Lógica Difusa, los cuales aportarán las bases de la funcionalidad del componente FIS. Atendiendo estas necesidades, el presente capítulo pretende mostrar un resumen, de fundamentos de Ingeniería de Software y la Lógica Difusa.

El capítulo comienza con los fundamentos de la Lógica Difusa, continúa con la descripción de la metodología de desarrollo de software utilizada y por ultimo se describen las estrategias de prueba de software implementadas. Los temas de Lógica Difusa ý de las metodologías de desarrollo de software, son tratados de manera resumida, si el lector requiere profundizar en estos temas, puede dirigirse a los libros citados en la bibliografía.

Lo expuesto en este documento acerca de Lógica Difusa, fue extraído en su gran mayoría del documento. Santiago Aja Fernández, Tesis doctoral “Un Nuevo Marco Matricial Para la Implementación de Inferencia Borrosa Aplicados al Procesado de Información no Numérica”, Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Escuela Técnica Superior de Ingenieros de Telecomunicaciones, Universidad de Valladolid, disponible en la biblioteca virtual Miguel de Cervantes. Las figuras 1, 2, 3, 4, 6,7 fueron extraídas de dicho documento.

2.2 LÓGICA DIFUSA Ó BORROSA.

La lógica borrosa es una rama de la inteligencia artificial que permite tratar con modos de razonamiento imprecisos. Es, en cierto modo, una extensión de la lógica bivaluada. Sus principios fueron establecidos por Lofti Zadeh,(profesor de Ingeniería Eléctrica en la universidad de California en Berkeley) a partir de los denominados conjuntos borrosos o difusos. Con la lógica borrosa, a diferencia de la clásica, es posible modelar los modos imprecisos de razonamiento que juegan un papel esencial en la habilidad del ser humano de tomar decisiones racionales en un entorno de incertidumbre e imprecisión.

2.2.1 Universo del discurso. El conjunto de valores numéricos que puede tomar para una variable discreta, o el rango de valores posibles para una continúa es lo que se conoce como Universo del Discurso de una variable X cualquiera.

2.2.2 Conjuntos borrosos o difusos. Desde el punto de vista formal el concepto de conjunto difuso es una generalización del concepto clásico de conjunto. La diferencia fundamental estriba en que, mientras que en la teoría clásica de conjuntos un determinado elemento puede pertenecer a un conjunto o no hacerlo, en la teoría de conjuntos difusos un elemento puede pertenecer a más de un conjunto con diferentes grados de pertenencia.

El **grado de pertenencia** es un número real dentro del intervalo [0, 1] que indica en qué proporción pertenece un determinado elemento a un conjunto. De este modo, si un elemento tiene un grado de pertenencia ‘0’ respecto a un conjunto dado, será equivalente a decir que dicho elemento no pertenece a dicho conjunto. Análogamente, si un elemento tiene un grado de pertenencia ‘1’, se dirá que dicho elemento se encuentra totalmente dentro del conjunto.

Formalmente un conjunto borroso se representa de la siguiente forma.

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

Donde:

A: es el nombre del conjunto. Puede ser cualquier letra mayúscula.

μ_A: es la función de membresía del conjunto, define el grado de pertenencia de un elemento al conjunto.

U: Universo del discurso.

Propiedades de los conjuntos borrosos. Las leyes y propiedades que cumplen los conjuntos clásicos no siempre se cumplen en el caso de los conjuntos borrosos. A continuación se analizará qué leyes verifican los conjuntos borrosos y cuales no.

- Propiedad conmutativa: Siempre se verifica, debido a que las t-normas y las t-conormas son conmutativas por definición.

- Propiedad asociativa: : Siempre se verifica, debido a que las t-normas y las t-conormas son asociativas conmutativas por definición

- Leyes de idempotencia: se cumplen si se elige el mínimo y el máximo como operadores para la intersección y la unión respectivamente. Pero si se escoge por ejemplo el producto algebraico y la suma algebraica no se cumple.

- Leyes de absorción: se cumplen si se elige el mínimo y el máximo como operadores para la intersección y la unión respectivamente. Con otras normas no ocurre necesariamente lo mismo

- Propiedad distributiva: se cumplen si se elige el mínimo y el máximo como operadores para la intersección y la unión respectivamente. Con otras normas no ocurre necesariamente lo mismo.

- Propiedad de absorción e identidad: siempre se cumplen por la última propiedad de t-norma y t-conorma.

- Involución del complemento: se cumple si el operador elegido es el estándar.

- Leyes de Morgan: Se garantiza su cumplimiento si la t-norma y la s-norma elegidas, derivan una de la otra. Es decir $T(x,y) = 1 - S(1-x,1-y)$.

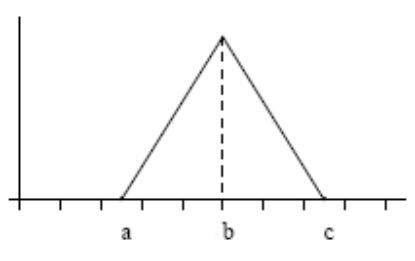
- Leyes complementarias: no se cumplen. Es quizás la consecuencia más clara de introducir la borrosidad a los conjuntos.

2.2.3 Función de pertenencia o membresía. También llamada función de inclusión.

$$\mu_A : U \longrightarrow [0,1]$$

La función de membresía es de tal modo que $\mu_A(x) \in [0,1]$ es el grado con el que un elemento $x \in U$ pertenece al conjunto borroso A. Cuando $\mu_A(x) = 0$ el elemento no pertenece al conjunto, y si $\mu_A(x) = 1$ pertenece totalmente. A continuación se presentan las funciones de membresía más comunes:

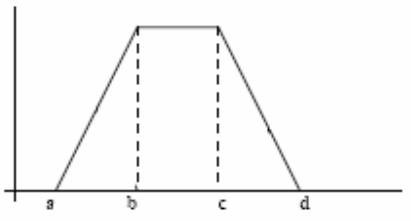
Tipo triangular. Una función de pertenencia triangular se determina a través de tres parámetros (a, b, c) de la siguiente manera:



$$Tr(x;a,b,c)=\begin{cases}0,x<a\\(x-b)/(a-b),a>x>b\\(c-x)/(c-b),b>x>c\\0,x>c\end{cases}$$

Los parámetros a, b, c determinan los tres ángulos de la función de pertenencia triangular, con $(a,b,c)>0 \quad y \quad a<b<c$.

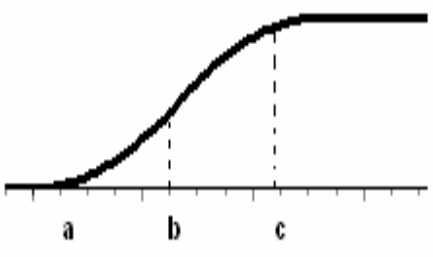
Tipo trapezoidal. La función de pertenencia trapezoidal se determina por cuatro parámetros (a, b, c, d) de la siguiente manera:



$$T(x;a,b,c,d)=\begin{cases}0,x<a\\(x-b)/(b-a),a>x>b\\1,b<x<c\\(c-x)/(d-c),d>x>c\\0,x>c\end{cases}$$

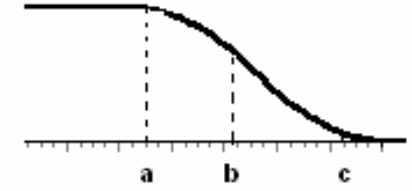
Los parámetros (a, b, c, d) determinan las coordenadas de los cuatro ángulos de la función de pertenencia trapezoidal, con $(a,b,c,d)>0 \quad y \quad a<b<c<d$

Tipo S. La función de pertenencia s se determina por tres parámetros (a, b, c) de la siguiente manera:



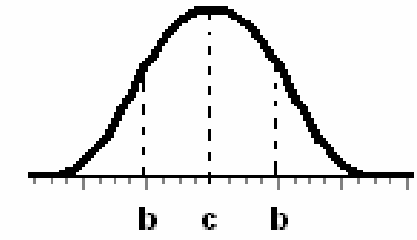
$$S(x;a,b,c)=\begin{cases}0, x<a\\ 2\left(\dfrac{x-a}{c-a}\right)^2, a>x>b\\ 1-2\left(\dfrac{x-a}{c-a}\right)^2, b<x<c\\ (c-x)/(d-c), d>x>c\\ 1, x>c\end{cases}$$

Tipo S invertida. La función de pertenencia s invertida se determina por tres parámetros (A, B, C) de la siguiente manera:



$$\pi(x;b,c)=\{1-S(x;c-b,(c-b)/2,c), x>c$$

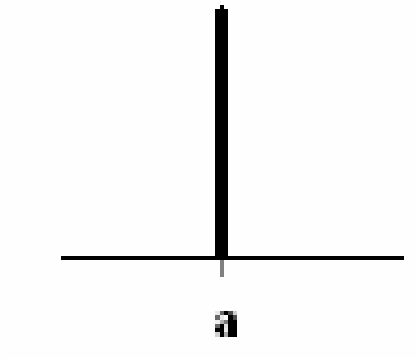
Tipo PI. La función de pertenencia pi se determina por tres parámetros (B, C) de la siguiente manera:



$$\pi(x;b,c)=\begin{cases}S(x;c-b,(c-b)/2,c), x<c\\ 1-S(x;c-b,(c-b)/2,c), x>c\end{cases}$$

b Representa el ancho de la campana y c el centro.

Tipo singleton. La función de pertenencia S invertida se determina por tres parámetros (a) de la siguiente manera:



$$f(x;a) = \begin{cases} 1, x = a \\ 0, x \neq a \end{cases}$$

2.2.4 Operaciones entre conjuntos borrosos. Las tres operaciones básicas definidas sobre los conjuntos clásicos (complemento, intersección y unión) pueden ser generalizadas a los conjuntos borrosos. Dentro la teoría de los conjuntos borrosos tiene especial relevancia la que hace uso de operaciones conocidas como operaciones estándar, definidas como:

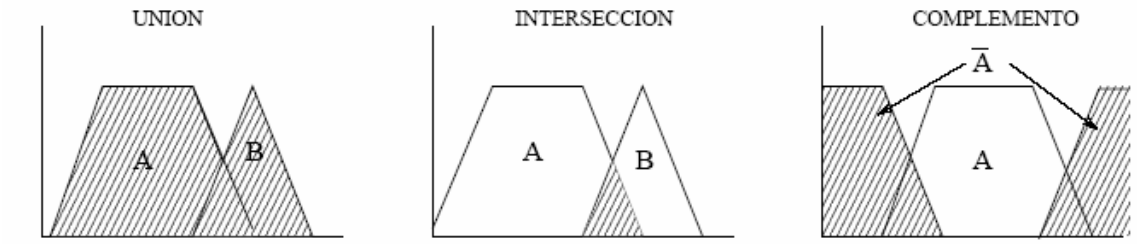
Intersección: $\mu A \bigcap B(x) = \min(\mu_A(x), \mu_B(x))$

Unión: $\mu A \bigcup B(x) = \max(\mu_A(x), \mu_B(x))$

Complemento: $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$.

No obstante esta no es la única forma posible de definir estas operaciones. Diferentes funciones pueden ser apropiadas para representarlas en diferentes contextos. En la figura 1 se puede ver la representación gráfica de estas operaciones.

Figura 1. Operaciones borrosas.



Complemento borroso. Dado un conjunto borroso $A \subset U$, se define su complemento como el conjunto borroso:

$\bar{A}$ Cuya función de pertenencia viene dada por la expresión

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

Donde C(x) es una función que debe cumplir las siguientes propiedades:

Condiciones de contorno: C (0)=1, C (1)=0.

Monotonía: para todo $a,b \in [0,1]$, si $a \leq b$ entonces $C(a) \geq C(b)$.

La función C(x) es conocida por algunos autores como c-norma.

En la mayoría de los casos, es deseable considerar algunos requerimientos adicionales para estas funciones.

1. C(x) es una función continua.
2. C(x) es involutiva lo que significa que C(C(a))=a $\forall a \notin [0,1]$

Existen muchas funciones que cumplen las propiedades antes descritas, algunas de estas son:

C(x)=1-x Estándar.

$C(x) = \left(\frac{1-x}{1-\lambda x} \right), \lambda \in (0,\infty)$ Sugeno.

$C(x) = \left(1 - x^w\right)^{1/w}$ Yager.

Intersección borrosa: T-NORMA. Dados dos conjuntos borrosos A y B, definidos sobre un mismo universo del discurso U, se define su intersección como un conjunto borroso $A \bigcap B$ cuya función de pertenencia viene dada por la expresión.

$$\mu A \bigcap B(x) = T[\mu_A(x), \mu_B(x)], \forall x \in U.$$

Donde la función T(x,y) es una norma triangular o t-norma. Una t-norma es una ampliación $T : [0,1]x[0,1] \to [0,1]$ T que verifica las siguientes propiedades:

1. Conmutativa: $T(x,y) = T(y,x), \forall x, y \notin [0,1].$
2. Asociativa: $\mathrm{T(T(x,y),z) = T(x,T(y,z)), \forall x,y,z \in [0,1]}$
3. Monotonía: $si(x \le y)$ y $(w \le z)$ entonces $T(x,w) \le T(y,z), \forall x, y, w \in [0,1].$
4. Elemento absorbente: $T(x,0) = 0, \forall x \in [0,1].$
5. Elemento neutro: $T(x,1) = x, \forall x \in [0,1].$

Existen muchas funciones que cumplen estas propiedades, por lo tanto pueden ser usadas para representar la intersección entre conjuntos borrosos. Algunas de estas son:

$T(x,y) = \min(x,y)$ Estándar.

$\mathrm{T(x,y) = \max(0, x + y - 1)}$ Diferencia acotada.

T(x,y)=x.y Producto algebraico.

En ocasiones es necesario restringir la posible t-norma considerando tres requerimientos adicionales.

1. Continuidad: T es una función continua.
2. Subidenpotencia: T(x, x) <x.
3. Monotonía estricta: $a_1<a_2$ y $b_1<b_2$ implica T (a_1, b1) <T (a_2, b2).

Es habitual encontrar en la bibliografía la intersección de dos conjuntos borrosos expresada como

$$\mu A\bigcap B(x) = \mu_A(x) \wedge \mu_B(x).$$

donde $\wedge$ representa el minino (inserción estándar).

Unión borrosa: T-CONORMA. dados dos conjuntos borrosos a y b, definidos sobre un mismo universo del discurso u, se define su unión como un conjunto borroso $A\bigcup B$ cuya función de pertenencia viene dada por la expresión.

$$\mu A\bigcup B(x) = S[\mu_A(x), \mu_B(x)], \forall x \in U.$$

Donde la función S(x,y) es una conorma triangular, también llamada t-conorma o s-norma. Es una aplicación $S:[0,1]x[0,1]\rightarrow[0,1]$ que satisface los siguientes requisitos:

1. Conmutativa: $S(x, y) = S(y, x), \forall x, y \notin [0,1].$
2. Asociativa: $S(S(x, y), z) = S(x, S(y, z)), \forall x, y, z \in [0,1]$
3. Monotonía: $si(x \leq y)$ y $(w \leq z)$ entonces $S(x, w) \leq S(y, z), \forall x, y, w \in [0,1].$
4. Elemento absorbente: $S(x,1) = 1, \forall x \in [0,1].$
5. Elemento neutro: $S(x,0) = x, \forall x \in [0,1].$

Existe un gran número de funciones que cumplen con estas propiedades, por lo cual se pueden utilizar para representar la Unión borrosa. Algunas de estas son:

$$\mathrm{S}(\mathrm{x},\mathrm{y}) = \max(x, y) \qquad \text{Estándar.}$$

$$S(x, y) = \min(1, x + y) \qquad \text{Suma Acotada.}$$

$$S(x, y) = x + y - x.y \qquad \text{Suma Algebraica.}$$

En ocasiones es necesario restringir la posible t-conorma considerando tres requerimientos adiciónales.

1. Continuidad: S es una función continua.
2. Subidenpotencia: S(x, x) > x.
3. Monotonía estricta: $a_1<a_2$ y $b_1<b_2$ implica $S(a_1,b_1)<S(a_2,b_2)$.

Es común habitual encontrar en la bibliografía la unión de dos conjuntos borrosos expresada como

$$\mu A \bigcup B(x) = \mu_A(x) \vee \mu_B(x).$$

donde $\vee$ representa el máximo (unión inserción estándar).

2.2.5 Principio de extensión. Proporciona un mecanismo para calcular los conjuntos borrosos obtenidos por medio de una transformación concreta (no borrosa) de cierto número (n) de conjuntos borrosos. Específicamente, si $X_{1,}X_2X_3,...,X_N$ son conjuntos borrosos con funciones de pertenencia $\mu_1(x_2), \mu_2(x_2), \mu_3(x_3),....,\mu_n(x_N)$ el nuevo conjunto $f(X_{1,}X_2X_3,...,X_N)$ tendrá como función de pertenencia:

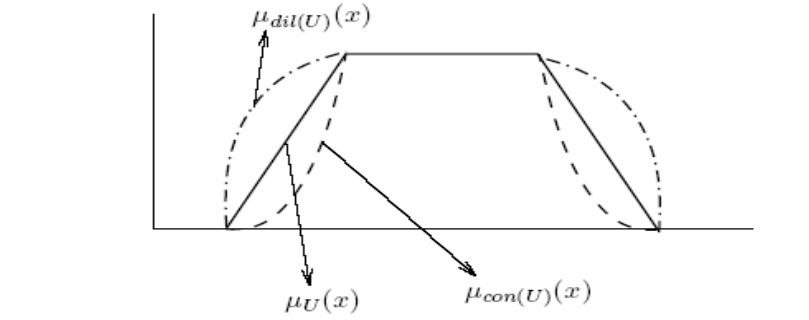
$$\mu(y) = \max_{x=f^{(-19}(y)} \left[\min_{i=1}^{n} \mu_i(x_i) \right]$$

2.2.6 Modificadores lingüísticos. Un modificador lingüístico, es una operación que modifica el significado de un término, o de una manera más general de un conjunto borroso. por ejemplo si un conjunto borroso denota *presión débil,* entonces *presión muy débil, presión más o menos débil, presión extremadamente débil* son ejemplos de modificadores aplicados a este conjunto. los modificadores se pueden ver como operadores que actúan sobre la función de pertenencia de un conjunto borroso para modificarla. algunos de estos operadores pueden ser:

1. Concentración: $\mu_{con(u)}(x) = [\mu U(x)]^2$.
2. Dilatación: $\mu_{dil(U)}(x) = [\mu U(x)]^{1/2}$
3. Modificadores artificiales: $u_{plus(u)}(x) = \left[\mu_U(x)\right]^{1.25}$
$\mu_{\min us(U)}(x) = \left[\mu_U(x)\right]^{0.75}$

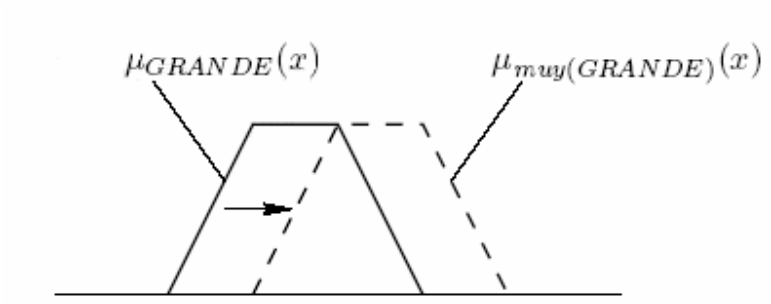
La figura 2 muestra el efecto de los dos primeros modificadores sobre un conjunto borroso con función de pertenencia trapezoidal.

Figura 2. Modificador Lingüístico (A)



La figura 3. Muestra el efecto del modificador muy sobre el conjunto borroso GRANDE.

Figura 3. Modificador Lingüístico (B)



2.2.7 Relación borrosa. Para dos universo del discurso U y V, una relación borrosa se define como un conjunto borroso R en el espacio UxV, cuya función de inclusión se denota como $\mu_R(u,v),$ con $u \in U$ y $v \in V$.

Producto cartesiano. Sean U y V dos universo del discurso cualquiera. Se define una relación borrosa R entre U y V como un conjunto borroso cuyo universo es el producto cartesiano UxV. Es decir:

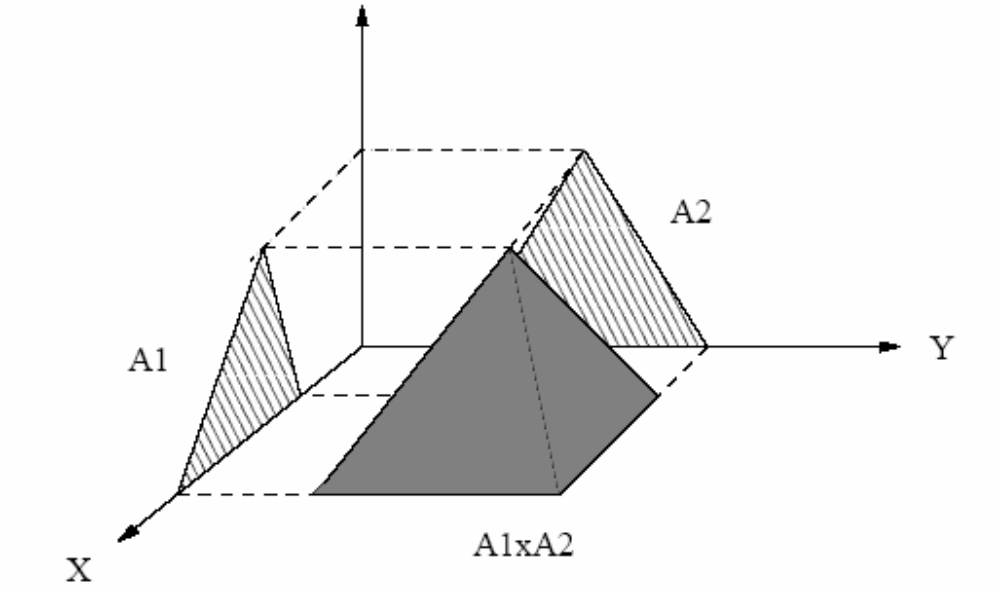
$$R = \{((x,y), \mu_R(x,y))/(x,y) \in UxV\}$$
$$\mu_R : [UxV] \rightarrow [0,1].$$

Ejemplo: Si $A_1 \subset$ U y $A_2 \subset V,$ y se define el producto cartesiano de A_1 y A_2 como:

$$\mu_{1}(A_1 xA_2)(x,y) = \min(\mu A_1(x), \mu A_2(y)).$$

La grafica de esta función se muestra en la figura 4.

Figura 4. Producto cartesiano



Operador conectivo "AND" ("Y"). Dados dos conjuntos borrosos A$\subset$U y B$\subset$V, y un par (x,y)$\in UxV,$ el conectivo "AND" que indica en qué medida x $\in$A e $y \in$B se puede implementar mediante la siguiente relación borrosa:

$$\mu AND(x, y) = \min(\mu_A(x), \mu_B(y)).$$

Se puede observar que está noción es muy parecida a la intersección de conjuntos borrosos. No es exactamente igual, ya que la operación de intersección se define para conjuntos en el mismo universo del discurso. Debido a esa similitud, es frecuente definir el conectivo "AND" mediante cualquier t-norma y no sólo mediante el empleo del mínimo.

Operador conectivo "OR ("O"). Dados dos conjuntos borrosos A$\subset$U y B$\subset$V, y un par (x,y)$\in UxV,$ el conectivo "OR" que indica en qué medida x $\in$A ó $y \in$B se puede implementar mediante la siguiente relación borrosa:

$$\mu OR(x,y) = \max(\mu_A(x), \mu_B(y)).$$

Al igual que el conectivo AND, el conectivo OR se puede implementar con cualquier t-conorma.

Operador implicación "THEN" ("ENTONCES"). De igual forma, se buscan relaciones para el operador de implicación "THEN" como relaciones borrosas entre antecedentes y consecuentes. En la Lógica Borrosa hay muchas maneras con las que puede definirse una implicación; basándose en dichas definiciones, pueden generarse muchas funciones de implicación diferentes en base a t-normas y t-conormas. A continuación se listan algunas de las formas de implementar la implicación.

$\mu M(x,y) = \min(\mu_A(x), \mu_B(y))$. Mamdani (mínimo).

$\mu p(x,y) = \mu_A(x).\mu_b(y)$. Larsen (Producto).

$\mu_R(x,y) = 1 - \mu_A(x) + \mu_A(x).\mu_B(y)$ Reichenbach.

$\mu_L(x,y) = \min(1 - \mu_A(x) + \mu_B(y), 1)$ Lucasiewick.

$\mu_W(x,y) = \max(1 - \mu_A(x), \min(\mu_A(x), \mu_B(y)))$ Willmott.

$\mu_{KD}(x,y) = \max(1 - \mu_A(x), \mu_B(y))$ Kleene-Dienes.

2.2.8 Variable lingüística. Una variable lingüística es una variable cuyos valores pueden expresarse mediante términos del lenguaje natural. Los diferentes términos o valores lingüísticos se representan mediante conjuntos difusos caracterizados por funciones de pertenencia definidas sobre su universo de discurso.

Formalmente, una variable lingüística se define por la tupla, (A, T(A), X, G, M) donde:

A representa el nombre de la variable.

T(A) conjunto de términos lingüísticos de A.

X es el universo de discurso de la variable A.

G regla sintáctica para general los términos lingüísticos.

M es una regla semántica que asigna a cada termino lingüístico t su Significado M (t), que es un conjunto borroso en A.

Ejemplo:

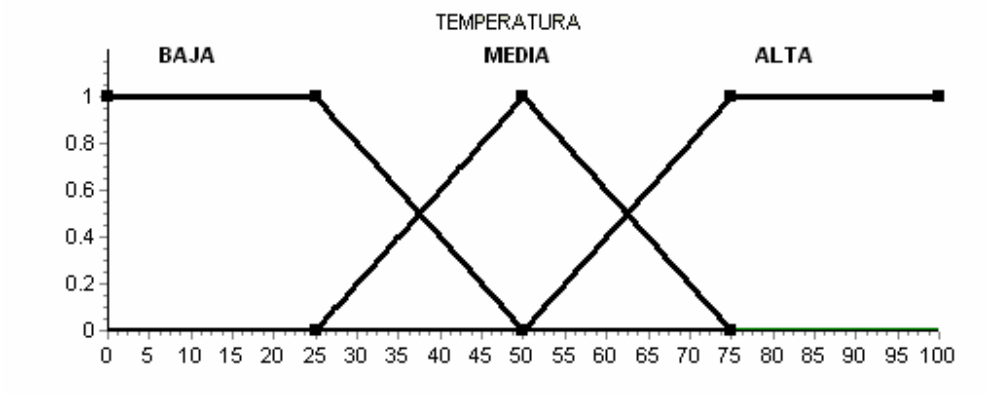
Temperatura puede considerarse como una variable lingüística, de modo que:

A Temperatura.

T (Temperatura) es el conjunto de todos los términos que pueden hacer referencia a temperatura como BAJA, MEDIA y ALTA entre otros.

El universo del discurso U va desde 0 hasta 100 grados centígrados. Ver figura 5.

Figura 5. Variable lingüística



2.2.9 Procedimientos de inferencia borrosa. Al contrario que en Lógica Clásica, el proceso de inferencia, en Lógica Borrosa no es preciso, sino que éste tiene lugar de una manera aproximada aunque el hecho no verifique la regla plenamente ("Razonamiento aproximado"). El razonamiento aproximado se resume generalmente, por extensión del razonamiento clásico, en los esquemas modus ponens generalizado y modus tollens genralizado, los cuales se presentan a continuación.

Proposición borrosa. En una proposición borrosa, a diferencia de lo que ocurre en las proposiciones clásicas, el grado de verdad puede estar en un valor intermedio entre uno y cero.

Existen varios tipos de proposiciones borrosa algunas de estas son:

- ✓ Incondicional y no cualificada: tiene la forma P: V es F. Ejemplo. P: Temperatura es Alta.
- ✓ Incondicional y cualificada: tiene la forma P: V es F es. Ejemplo Hessmann es joven es muy cierto.
- ✓ Condicional y no Cualificada: tiene la forma P: Si X es A, Entonces (Then) Y es B. Ejemplo. SI Temperatura es Alta Entonces Ventana Es Abierta.
- ✓ Condicional y Cualificada: Tiene la forma P: Si X es A, Entonces (Then) Y es B es S.

Condicional y no Cualificada. Tiene la forma P: Si X es A, Entonces (Then) Y es B. Donde X e Y son variables con valores en los universos U y V respectivamente y A y B son conjuntos borrosos sobre los U y V respectivamente. Esta expresión se puede escribir más formalmente por medio de la relación.

<X, Y> es R

Donde R es un conjunto borroso definido en UxV, y su función de pertenencia de cada punto (x,y)$\in$ UxV por medio de una *implicación borrosa*

$$R(x,y) = \tau(\mu_A(x), \mu_B(y))$$

La implicación se puede definir por cualquiera de las funciones ya descritas.

Modus ponens generalizado. En este esquema, se expresa una regla, se da un hecho y a partir de ambos se obtiene una conclusión.

Regla: **Si X es A entonces Y es B**
Hecho: **X es A'**
Conclusión: **Y es B'**

El valor de B' se obtiene aplicando el operador implicación correspondiente

$$\mu B'(y) = \tau(\mu A(A'), \mu B(y)) .$$

Modus tollens generalizado. En el modus tollens generalizado se cumple que:

Regla: **Si X es A entonces Y es B**
Hecho: **Y es B'**
Conclusión: **X es B'** **(MTG)**

2.2.10 Razonamiento aproximado multicondicional. Es te razonamiento es una generalización del modus ponens a múltiples condiciones

Regla: **Si X es A_1, entonces Y es B_1**
Regla: **Si X es A_2, entonces Y es B_2**
…… **…** **…**
Regla: **Si X es A_n, entonces Y es B_n**
Hecho: **X es A'**
Conclusión: **Y es B'**

Esta clase de problemas de inferencia se pueden manejar también a partir de lo que se conoce como el *método de interpolación,* que consiste en dos pasos:

Encontrar el grado de consistencia del antecedente A_j, $j \in \{1,.....,n\}$ con el hecho

A’

$$\text{dcj(A)} = h(A' \bigcap A_j)$$

Siendo h () una medida de la altura del conjunto, que puede calcularse usando una t-norma

$$\text{dcj(A)} = \bigcup_{x \in U} (A'(x) \bigcap A_j(x))$$

Usando las operaciones estándar resulta

$$\text{dcj(A)} = \max_{x \in U}[\min[(A'(x), A_j(x)]]$$

Calcular la conclusión B’ truncando cada conjunto B_j por el valor r_j(A’) y tomar la unión de los conjuntos truncados.(en casos de que las reglas sean disyuntivas)

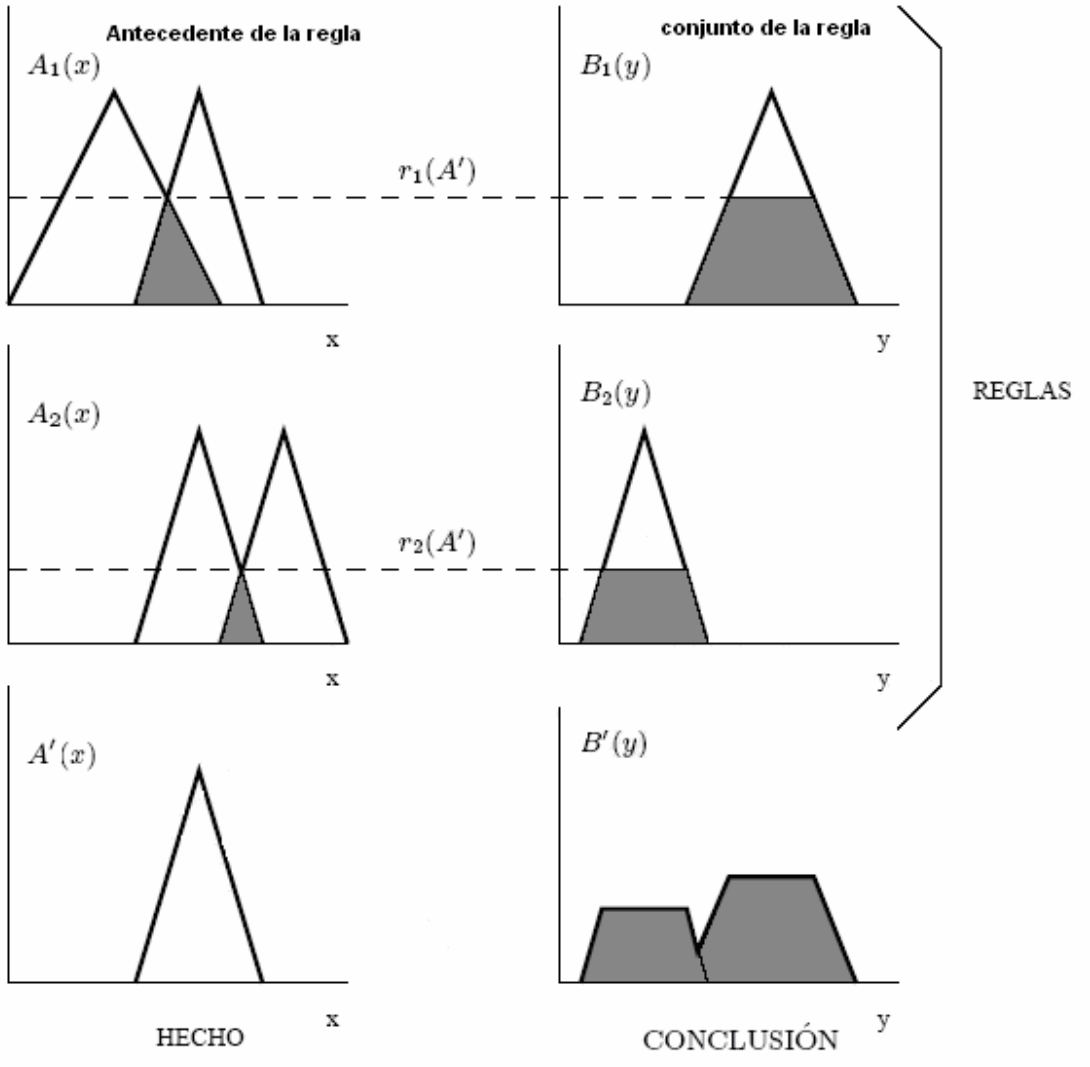
$$B'(y) = \bigcup_j \left(dc_j(A') \bigcap B_j(y)\right) \forall y \in Y$$

Usando las operaciones estándar resulta ser

$$B'(y) = \max_j \left[\min\left(dc_j(A'), B_j(y)\right)\right] \forall y \in Y$$

En la grafica 6 se puede ver el proceso.

Figura 6. Razonamiento aproximado multicondicional



Se puede observar que el valor de $r_j(A')$ es igual a la implicación entre el valor del antecedente de la regla y el conjunto de salida. Para el caso de la grafica operador de la implicación es el mínimo.

2.2.11 Borrosificador. el borrosificador establece una relación entre puntos de entrada no borrosos al sistema $x=(x_1, x_2,\ldots, x_n)^t$, y sus correspondientes conjuntos borrosos. se pueden utilizar diversas estrategias de borrosificación.

Borrosificador singleton. Es el método más utilizado, principalmente en sistemas de control, y consiste en considerar los propios valores discretos como conjuntos borrosos. De otra forma, para cada valor de entrada **x** se define un conjunto A' que lo soporta, con función de pertenencia $\mu_A(x')$, de modo que $\mu_A(x)=1$, **(x'=x), y** $\mu_A(x')=0$, para todo **x** $\in$ **U** en los que **x'** $\neq$ **x.**

Se pueden definir borrosificadores utilizando cualquiera de las funciones membresía descritas anteriormente. Como por ejemplo utilizar funciones tipo Pi.

Para los borrosificadores no singleton deben definirse puntos de evaluación que el número de intervalos en los cuales se dividirá la función que representa el difusor.

Para calcular el grado de pertenencia de un elemento a un conjunto borroso A Gp_A con función de pertenencia μ_A, de una variable lingüística, cuyo difusor D tiene función de pertenencia μ_D y N puntos de evaluación. Existe diferentes métodos, a continuación se presentan algunos.

- ✓ **Método Max-Min :** $Gp_A = \max\left[\min_{i=0}^{N}\left(\mu_A(x_i), \mu_D(x_i)\right)\right]$**.**
- ✓ **Método Sup-Produc:** $Gp_A = \frac{1}{k}\sum_{i=0}^{N}\mu_A(x_i)\,\mu_D(x_i)$**.** Donde K es un factor de normalización.

- ✓ **Método Max-Produc:** $Gp_A = \frac{1}{K}\max_{i=0}^{N}\left[\mu_A(x_i), \mu_D(x_i)\right]$ **.** Donde K es un factor de normalización.
- ✓ **Método Sum-Min:** $Gp_A = \frac{1}{K}\sum_{i=0}^{N}\min(\mu_A(x_i), \mu_D(x_i))$**.** Donde K es un factor de normalización.

2.2.12 Sistemas de inferencia borrosa (FIS). Un sistema de inferencia borrosa es la aplicación de la inferencia borrosa a la automatización de procesos.

Tipos de sistemas basados en reglas borrosas. En la literatura especializada se distinguen tres clases se distinguen tres clases de sistemas basados en reglas borrosas, de acuerdo con la forma de las reglas y del tipo de entradas y salidas.

Sistemas puros. Estos sistemas tienen como entrada y como salida conjuntos borrosos. Al no realizar ninguna transformación sobre las entradas o sobre las salidas, tienen sólo dos componentes principales: una base de conocimiento y un motor de inferencia.
Las reglas lingüísticas empleadas son de la forma.

SI X_1 es A_1 y….X_n es A_n ENTONCES T ES B

Donde X_i e Y son variables lingüísticas, y los A_i y B son etiquetas lingüísticas asociadas a conjuntos borrosos.

Sistemas borrosos tipo mamdani. este tipo de sistemas fue propuesto por mamdani, quien fue capaz de traducir las teorías borrosas propuestas por zadeh en el primer sistema borroso aplicado a un problema de control. esta

clase de sistemas, es la más usada dentro de los sistemas borrosos, se conoce también con el nombre de *controladores borrosos.*

Un sistema tipo Mamdani se corresponde con la noción más conocida de *sistemas borrosos;* está compuesto por una base de conocimiento, un motor de inferencias, y unas interfaces de borrosificación y desborrosificación. Estos sistemas tienen una serie de características interesantes:

- ✓ Pueden ser usados en aplicaciones del mundo real, ya que manejan con facilidad entradas y salidas reales.
- ✓ Proporcionan un marco natural para la inclusión de conocimiento experto en forma de las reglas lingüísticas.
- ✓ Tienen gran libertad a la hora de elegir las interfaces de borrosificación y desborrosificación.

Por el contrario presentan también una serie de limitaciones.

- ✓ Falta de flexibilidad en el sistema basado en reglas borrosas debido a la forma tan rígida en la que se dividen los espacios de entradas y salida.
- ✓ No existe una distinción clara entre el conocimiento experto y la definición de las variables lingüísticas incluidas en las reglas borrosas.
- ✓ Cuando la las variables de entrada al sistema dependen unas de otras, es muy complicado obtener una partición borrosa adecuada de los espacios de entrada.
- ✓ El tamaño de la base de conocimiento depende directamente del número de variables y conjuntos borrosos que existan en el sistema.

Las reglas que maneja esta clase de sistemas son de la forma

SI X_1 es A_1 y….X_n ENTONCES Y es B

Donde las entradas X_i y la salida Y son ahora números (no borrosos), en lugar de términos lingüísticos (como los sistemas puros), y por lo tanto los A_i y B son conjuntos borrosos sin interpretación directa, en lugar de etiquetas lingüísticas.

A continuación se describen los elementos de este sistema de inferencia borroso.

- **Motor de Inferencias:** que como su nombre lo indica inferirá las acciones del sistema empleando alguna representación de la implicación borrosa, así como de los procedimientos de inferencia en Lógica Borrosa.

- **Interfaz de borrosificación:** se encarga de tomar las entradas al sistema (por lo general no borrosas) y convertirlas en valores borrosos

- **Base de conocimiento:** contiene las reglas que gobierna el sistema. Estas reglas tienen la forma SI ENTONCES, con el antecedente relacionado con el conectivo Y por lo general.

- **Desborrosificador:** El desborrosificador es la función que transforma un conjunto borroso de universo de discurso U, normalmente de salida de un dispositivo de inferencia borrosa. Para esta tarea se utilizan diversos métodos.

✓ **Método del centroide o centro de gravedad:** esta definido por la expresión

$$y = \frac{\int y\mu(y)dy}{\int \mu(y)dy}$$

Para el caso de trabajar con valores discretos, la expresión queda

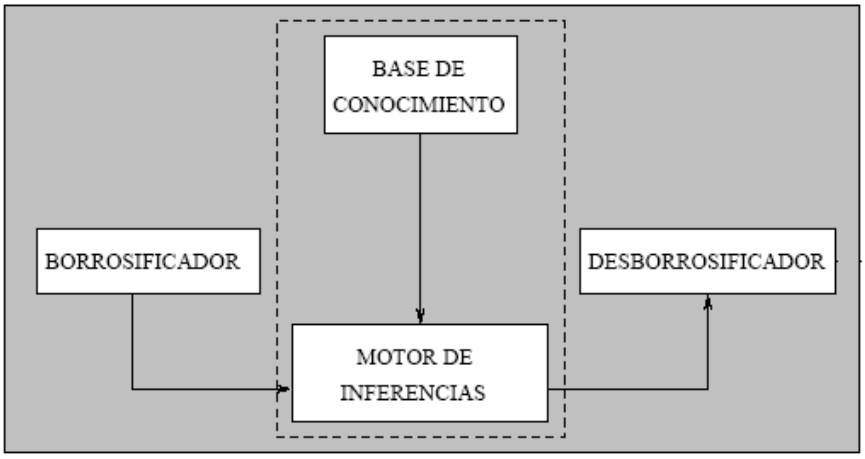
$$y = \frac{\sum_{k=1}^{N} y_k \mu(y_k)}{\sum_{k=1}^{N} \mu(y_k)}$$

- ✓ **Método singleton (semifallo).** Consiste en calcular el promedio de los centroides de las funciones de los conjuntos de salida activados. Para la ponderación se utilizan los pesos de las reglas activadas, o grados de pertenencia de la salida a los subconjuntos correspondientes.
- ✓ **Método media de centros:** se define como

$$y = \frac{\sum_{i=1}^{N} y^{-l} \mu_{B'}(y)}{\sum_{i=1}^{N} \mu_{B'}(y)}$$

Donde y^{-l} representa el centro del conjunto B' (definido como el punto donde el conjunto B' alcanza el valor máximo) y $\mu_{B'}(y)$ es el grado de activación de la regla. La figura 7 muestra el esquema de un sistema Mandani.

Figura 7. Esquema de sistema Mandani.



Sistemas borroso tipo TAKAGI-SUGENO-KANG. en lugar de trabajar con las reglas lingüísticas como las de la sección anterior, takagi, sugeno y kang propusieron un nuevo modelo basado en reglas donde el antecedente estaba compuesto de variables lingüísticas y el consecuente se representaba con una función de las variables de entrada. la forma más habitual de esta clase de es la siguiente:

SI X_I es A_1 y …. X_n ENTONCES $Y=p_1X_1+p_2X_2+…+p_nX_n+p_0$

Siendo X_{1i} las variables de entrada, Y la variable de salida, y p_i parámetros reales. Esta clase de reglas se conoce como reglas TSK en referencia a sus creadores.

La salida de un sistema borroso TSK que usa una base de conocimiento con m reglas se obtiene como la media ponderada de las salidas individuales proporcionadas por cada regla, Y_i, (i=1,…..,m) , como sigue

$$Salida = \frac{\sum_{i=1}^{m} h_i Y_i}{\sum_{i=1}^{m} h_i}$$

Siendo $h_i = T(A_1^i(x_i),....,A_n^i(x_n))$ el grado de emparejamiento entre la perte antecedente de la regla i y las entradas actuales al sistema x=(x_1,…,x_n). T es un operador de conjunción que se modela mediante una t-norma.

2.3 TEORÍA DEL DESARROLLO E IMPLEMENTACIÓN DE SOFTWARE.

Para iniciar el desarrollo del software, es necesario definir un marco de referencia común. Este marco de referencia guiará él proceso definiendo pasos, actividades y tareas a desarrollar conducentes a la elaboración de

productos con determinadas características. Este marco de referencia se conoce como ciclo de vida de desarrollo de software, el cual abarca la vida del sistema desde su concepción hasta la finalización de su desarrollo.[12]

Existen muchos modelos de ciclo de vida que permiten orientar el desarrollo de aplicaciones entre los cuales hay que mencionar: modelo espiral, espiral WINWIN, desarrollo concurrente, cascada o lineal secuencial y modelos orientados a objetos. Para el caso de este proyecto de grado se eligió el modelo de construcción de prototipos, también conocido como prototipos evolutivos. A continuación se presenta una explicación de los fundamentos de este modelo y las razones por la cual se eligió.

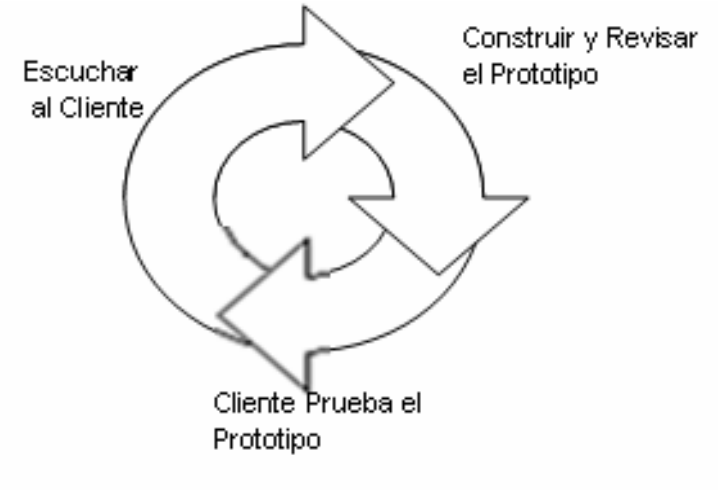
2.3.1 Modelo de construcción por prototipos[13]. Esta metodología de desarrollo se utiliza cuando el cliente define un conjunto de objetivos generales para el software pero no identifica los requisitos detallados de entrada, proceso o salida. también, cuando el responsable del desarrollo de software no tiene claro la manera de cómo va a desarrollar la solución a los objetivos generales que solicita el cliente.

El proceso comienza con la recolección de requisitos, donde el desarrollador y el cliente se encuentran y definen los objetivos globales para el software, se identifican algunos requisitos conocidos y los aspectos donde es obligatoria una profundización. Entonces aparece un diseño “rápido”, el cual se centra en una representación de esos aspectos del software que serán visibles para el usuario/cliente, por ejemplo: enfoques de entrada y parámetros de salida. El diseño rápido lleva a la construcción de un prototipo, para lo cual se utiliza cualquier metodología de desarrollo de software, para el caso de este

[12] PIATTINI Mario G *etal*. Análisis y Diseño de Aplicaciones Informáticas de Gestión. Ciclo de vida del Software. p.39.
[13] PRESSMAN, Op. cit, p. 21.

proyecto, las metodologías orientadas a objetos. El prototipo lo evalúa el usuario/cliente para refinar los requisitos del software a desarrollar (Figura 8).

Figura 8. Paradigma de Construcción de Prototipos.



La iteración ocurre cuando el prototipo se pone a punto para satisfacer las necesidades del cliente, permitiendo al mismo tiempo que el desarrollador comprenda mejor lo que se necesita hacer.

Esta metodología de desarrollo, parece ser perfecta, pero como nada en la vida es perfecto, a continuación se presentan sus principales desventajas:

- El cliente ve lo que parece ser una versión de trabajo del software, sin saber que con la prisa de hacer que algo funcione no se ha tenido en cuenta la calidad del software global o la facilidad de mantenimiento a largo plazo.

- El desarrollador, a menudo, hace compromisos de implementación para hacer que el prototipo funcione rápidamente. Se puede utilizar un sistema

operativo o lenguaje de programación inadecuado simplemente porque está disponible y por que es conocido; un algoritmo eficiente se puede implementar simplemente para demostrar la capacidad.

Este proyecto de grado, al ser planteado, tenía claro cuales eran los datos entrada y salida, además se tenía definido cual iba ser la manera de generar dichas salidas. Pero no se tenía la forma de comunicación entre el componente FIS y EVOLUCION. Otro aspecto en el cual no había suficiente claridad era en la interfaz grafica o interacción con el usuario. Por estas razones se decidió utilizar este enfoque de desarrollo, por que el desarrollo de prototipo le permite al cliente y a los desarrolladores resolver las dudas y requisitos del software. En plan de proyecto se planteo el modelo de desarrollo en espiral, pero se desistió de este al notar que los requisitos generales no cambian. Sabiendo que el modelo en espiral es adecuado cuando los requisitos del sistema cambian.

Metodología de desarrollo14. Una vez definido el marco de referencia que guiará el proceso de desarrollo, se debe definir la metodología que va seguir este proceso de construcción, es decir, se debe especificar que filosofía, acciones, fases, reglas, técnicas, herramientas se van a ejecutar en cada una de las etapas y que permitirán detallar como se obtienen los productos, en este caso los prototipos, que se definieron en el ciclo de vida escogido**.**

Las metodologías de desarrollo especifican:

- ✓ Como se debe dividir un proyecto en fases.
- ✓ Qué tareas se llevan a cabo en cada fase.
- ✓ Qué salidas se producen y cuando se deben producir.
- ✓ Qué restricciones se aplican.

[14] PIATTINI. Metodologías de Desarrollo de Software, Op. cit, p. 61-62.

- ✓ Qué herramientas de van a utilizar.
- ✓ Como se gestiona y controla un proceso.

Existen varias metodologías que facilitan el desarrollo de sistemas, las dos más importantes son:

Metodología de desarrollo estructurado: proponen la creación de modelos del sistema que representen los procesos, los flujos y la estructura de datos de una manera descendente. Esta visión se puede enfocar en las funciones del sistema, en la estructura de los datos o en ambos aspectos. Dando lugar a los siguientes tipos de metodologías: orientadas a procesos, orientadas a datos (jerárquicas y no jerárquicas) y mixtas.

En esta metodología se produce una división de los elementos que constituyen un sistema: funciones que llevan a cabo los programas y los datos que se almacenan en archivos o base de datos.

Metodologías orientadas al objeto: el dominio del problema se caracteriza por mediante un conjunto de objetos con atributos y comportamientos específicos. Los objetos son manipulados mediante una colección de funciones (llamadas métodos, operaciones o servicios) y se comunican entre ellos mediante un protocolo de mensajes. Los objetos se clasifican mediante clases y subclases.

Para este proyecto, se eligió la metodología orientada al objeto porque el software al cual se le integrará (EVOLUCION 3.5) el componente FIS está desarrollado bajo esta filosofía. Además de las múltiples ventajas que esta ofrece.

El paradigma orientado a objeto[15]**.** El proceso orientado a objetos se mueve a través de una espiral evolutiva que comienza con el usuario. es aquí donde se define el dominio del problema. la planificación y el análisis de riesgos establecen una base para el plan del proyecto orientado a objetos. la ingeniería del software orientada a objetos hace hincapié en la reutilización, por lo tanto, las clases se buscan en una biblioteca (de clases orientada a objetos) antes de construirse. cuando una clase no puede encontrarse en la biblioteca, el desarrollador de software aplica el análisis orientado a objetos, diseño orientado a objetos, programación orientada a objetos para crear la clase y los objetos derivados de la clase. la nueva clase se pone en la biblioteca de tal manera que pueda ser reutilizada a futuro.

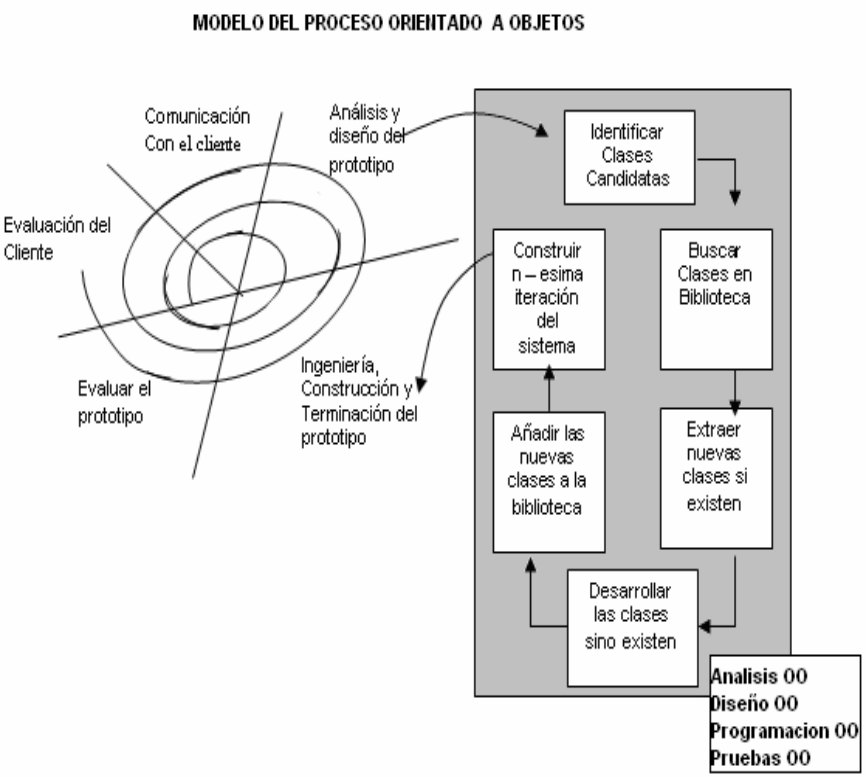
La figura 9 muestra el proceso de desarrollo orientado a objetos.

- Ventajas del paradigma orientado a objetos[16]: un objeto encapsula tanto datos como los procesos que se aplican a esos datos. Esta importante característica permite construir clases de objetos e inherentemente construir bibliotecas de objetos y clases reutilizables.

El paradigma de orientación a objetos es tan atractivo para tantas organizaciones de desarrollo de software debido a que la reutilización es una atributo importantísimo en la ingeniería del software (reduce costos y tiempos de desarrollo). Además, los componentes de software derivados mediante el paradigma de objetos muestran características (como la independencia funcional, la ocultación de información, etc.) asociados al software de alta calidad.

[15] PRESSMAN, Op. cit, p. 344.
[16] PRESSMAN, Op, cit, p. 343.

Figura 9. PRESMAN, Op. Cit, p. 344.



Desarrollo modular[17]**:** El Software Se Divide En Componentes Nombrados Y Abordados Por Separado, Llamados Frecuentemente *Módulos*, Que Se Integran Para Satisfacer Los Requisitos Del Problema.

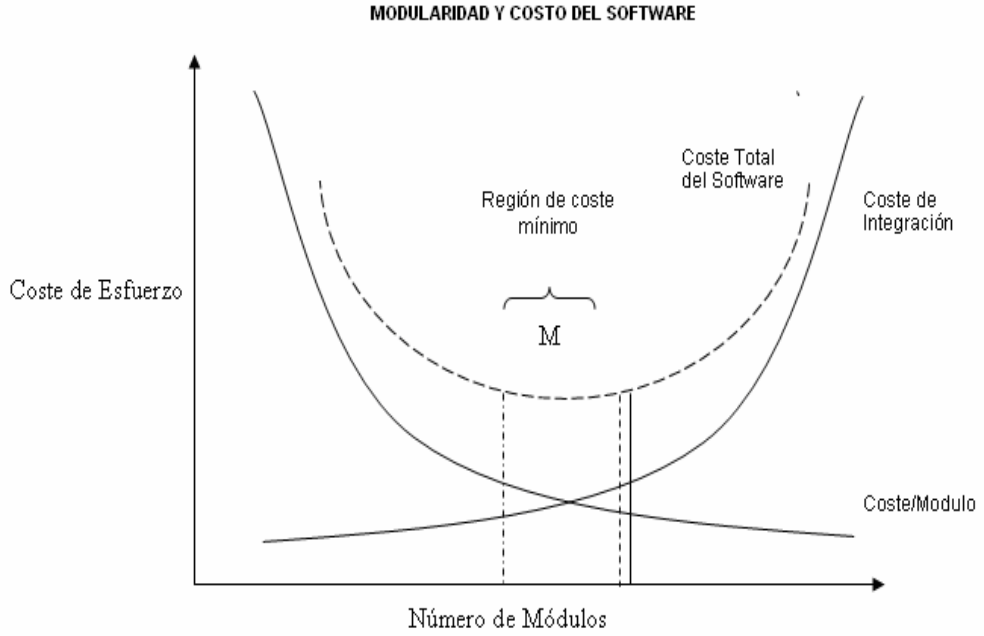
Es posible pensar que si se subdivide el software indefinidamente, el esfuerzo que se requiere para desarrollarlo sería mínimo. Desgraciadamente, intervienen otras fuerzas, que hacen que esta conclusión sea falsa. Como muestra la figura 10, el esfuerzo (coste) para desarrollar un módulo de software individual disminuye a medida que aumenta el número total de módulos. Dado el mismo conjunto de requisitos, tener más módulos conduce

[17] PRESSMAN, Op, cit, p. 224.

a un tamaño menor de módulo. Sin embargo, a medida que aumenta el número de módulos, también crece el esfuerzo (coste) asociado con la integración de módulos. Estas características conducen también a la curva total de coste o esfuerzo que se muestra en la figura.

Las curvas que se muestran en la figura 10 proporcionan en efecto una guía útil cuando se tiene en consideración la modularidad. Esta debe aplicarse, pero teniendo cuidado de estar próximo a M.

Figura 9. PRESMAN, Op. Cit, p. 225.



Para realizar este proyecto de grado se utiliza un modelo de desarrollo orientado a producir prototipos software, para la construcción de cada prototipo se usa una combinación entre la metodología orientada a objetos y un desarrollo modular. La metodología orientada a objetos, permite mirar el problema como un sistema y modelar su estructura y comportamiento a

través de clases. Para abordar la complejidad del problema, se emplea un desarrollo modular que permite descomponerlo en subproblemas reduciendo en gran medida la complejidad inherente al sistema y así conseguir una solución efectiva.

El proceso de diseño[18]**:** el diseño de software es un proceso iterativo mediante el cual requisitos se traducen en un "plano" para construir el software. inicialmente, el plano representa una visión holística del software. esto es, el diseño se representa a un nivel alto de abstracción. a medida que ocurren las iteraciones del diseño, refinamiento subsiguiente conduce a representaciones de diseño a niveles de abstracción mucho más bajos. estos niveles podrán rastrear aún según los requisitos, pero la conexión más sutil.

Para expresar el diseño se utiliza UML (Lenguaje Unificado de modelado) por dos razones:

1. El diseño de EVOLOCION 3.5 esta expresado utilizando este lenguaje.
2. UML se está convirtiendo en el estándar de la industria para representar el diseño de sistemas que involucren software.

Pruebas orientadas a objetos[19]**:** El objetivo de las pruebas, es encontrar el mayor número posibles de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista. A pesar que este objetivo fundamental permanece inalterado para el software orientado a objetos, la naturaleza de los programas orientados a objetos cambian las estrategias y tácticas de prueba.

[18] PRESSMAN, Op, cit, p. 221.
[19] PRESSMAN, Op, cit, p. 407.

Los métodos de diseño de casos de prueba para software orientado a objetos de contemplar mínimo los siguientes aspectos:

1. Cada caso de prueba debe ser identificado separadamente, y explícitamente asociado con la clase a probar.
2. Debe declararse el propósito de la prueba.
3. Debe desarrollarse una lista de pasos a seguir, como consecuencia de la prueba, pero además debe contener:

- ✓ Definición de una lista de estados, específicos para el objeto a probar.
- ✓ Una lista de mensajes y operaciones, que se ejercitarán como consecuencia de las pruebas.
- ✓ Una lista de excepciones, que pueden ocurrir conforme al objeto que se comprueba.
- ✓ Una lista de condiciones externas. Por ejemplo, los cambios en el ambiente externo al software, que debe existir para conducir apropiadamente las pruebas.
- ✓ Información adicional, que ayudará a la comprensión e implementación de la prueba.

A diferencia del diseño de pruebas convencional, que se conduce mediante una visión entrada-salida de software, o detalle algorítmico de los módulos individuales, la prueba orientada a objetos se enfoca en las secuencias de operaciones de diseño apropiadas para probar los estados de una clase.

Factores de calidad ISO 9126[20]. El estándar ISO 9126 ha sido desarrollado en un intento de identificar los atributos claves de calidad para el software. El estándar identifica seis atributos claves de calidad:

[20] PRESSMAN, Op, cit, p. 326.

- **Funcionalidad:** el grado en que el software satisface las necesidades indicadas por los siguientes subatributos: idoneidad, corrección, interoperatividad, conformidad y seguridad.

- **Confiabilidad:** cantidad de tiempo que el software está disponible para su uso. Está referido por los siguientes subatributos: madurez, tolerancia a fallos, y facilidad de recuperación.

- **Usabilidad:** Grado en que el software es fácil de usar. Viene reflejado por los siguientes subatributos: facilidad de compresión, facilidad de aprendizaje, y operatividad.

- **Eficiencia:** grado en que el hace óptimo uso de los recursos del sistema. Está indicado por los siguientes subatributos: tiempo de uso y recursos utilizados.

- **Facilidad de mantenimiento:** la facilidad con que una modificación puede ser realizada. Está indicada por los siguientes subatributos: facilidad de análisis, facilidad de cambio, estabilidad y facilidad de prueba.

- **Portabilidad:** la facilidad con que el software puede ser llevado de un entorno a otro. Está referido por los siguientes subatributos: facilidad de instalación, facilidad de ajuste, facilidad de adaptación al cambio.

3. FASE DE FORMACIÓN

Esta fase del proyecto giró entorno a cuatro problemas:

1. Clarificar el objetivo general y los objetivos específicos del proyecto.
2. Aprender lógica borrosa o difusa.
3. Formarse en el desarrollo de software orientado a objetos, utilizando UML como medio para expresar los planos del software y lenguaje programación DELPHI 7 como herramienta de desarrollo.
4. Familiarizarse con el diseño e implementación de EVOLUCION 3.5, para lograr realizar la integración entre EVOLUCION 3.5 y el componente FIS.

Para resolver problema de clarificar el objetivo general y los objetivos específicos del proyecto se realizaron las siguientes actividades:

- Reuniones precedidas por el director de proyecto y los Ingenieros de Sistemas César Augusto Rivera Palacios, Freddy Sarmiento Villamizar. El objetivo de estas reuniones era expresar la forma de integrar la lógica borrosa o difusa a la Dinámica de Sistemas, en particular, como tendría que trabajar un elemento, dentro de EVOLUCION 3.5, definido mediante un FIS[21].

Con el fin de aprender lógica borrosa se hizo lo siguiente:

- El señor Cesar Eduardo Gonzáles Pérez, matriculó la materia Técnicas Adaptativas para el Razonamiento Aproximado con el Ingeniero Juan Carlos Reyes. Lo cual le permitió aprender los conceptos básicos del tema y el

[21] FIS Sistema de Inferencia Borrosa o Difusa.

manejo del toolbox para trabajar lógica borrosa de matlab (fuzzy logic). Con esto se logró tener una idea de cuales tendrían que ser las funciones básicas del componente FIS.

- Se revisó bibliografía sobre lógica borrosa, consultando por intermedio de bibliotecas de las principales universidades de Bucaramanga. Al terminar este ejercicio se tomo la decisión de adquirir dos libros con los cuales profundizar y afianzar los conocimientos en el tema, los libros son: ***REDES NEURONALES Y SISTEMAS DIFUSOS, 2ª EDICIÓN, BONIFACIO MARTÍN DEL BRÍO, ALFREDO SANZ MOLINA, ALFAOMEGA RA-MA*** y ***FUZZY SETS AND FUZZY esbo THEORY AND APPLICATIONS, GEORGE J. KLIR, BO YUAN, PRENTICE HALL.*** Este ultimo disponible en la colección general de la biblioteca de la **UNIVERSIDAD INDUSTRIAL DE SANTANDER.**

- Luego de explorar en Internet se llegó a la conclusión, que la información disponible sobre el tema, se encuentra muy dispersa, es muy superficial en su gran mayoría y repetitiva. Por tal motivo el único articulo que se asumió como soporte fue: ***Santiago Aja Fernández, Tesis doctoral Un nuevo marco matricial para la implementación de inferencia borrosa aplicados al procesado de información no numérica, Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Escuela Técnica Superior de Ingenieros de Telecomunicaciones, Universidad de Valladolid***. Disponible en la biblioteca virtual Miguel de Cervantes.

Para llevar a cabo la formación en el desarrollo de software orientado a objetos, utilizando UML como medio para expresar los planos del software y lenguaje programación DELPHI 7 como herramienta de desarrollo del mismo. Se realizaron las siguientes actividades:

- En el campo de la ingeniería del software se aprovechó la experiencia adquirida al ver la materia Ingeniería del Software, con el Ingeniero Javier Fernández. Además se estudiaron los siguientes libros.

- Ingeniería del software un enfoque práctico. Roger S. Presuman, Mc Graw Hill.

Para suplir las falencias en desarrollo de software orientado a objetos, se realizaron charlas con el Ingeniero Emiliano Lince Mercado y en complemento se estudió el libro:

- Construcción de software orientado a objetos. 2 *ed.* Madrid, MEYER, Bertrand. Prentice Hall.

- Diseño Orientado a Objetos. Grady Booch, Traducción Jaime Octavio Albarracín. Universidad Industrial de Santander.

Estudiar UML utilizando los libros:

- El Lenguaje Unificado de Modelado, Grady Booch, James Rumbaugh, Ivar Jacobso. Addison Wesley.

En el problema de aprender el leguaje de programación DELPHI 7, se utilizaron los siguientes libros:

- Programación en turbo pascal versiones 5.5, 6.0 y 7.0. 2ª edición. Luís Joyanes Aguilar. Mc Graw Hill.

- Guía de desarrollo DELPHI 5. Steve Teixeira, Xavier Pacheco. Prentice Hall.

Para la tarea de estudiar el diseño y código fuente EVOLUCION, se realizaron básicamente dos Actividades.

- Charlas con el Ingeniero Emiliano Lince Mercado, quien es uno de los desarrolladores de EVOLUCION 3.5.

- Estudiar los diagramas UML y el manual de programador de EVOLUCION 3.5.

Además de las actividades anteriormente planteadas, se estudiaron el software UNFUZZY[22] 1.2 y la herramienta para Lógica Difusa de MatLab. Este estudio permitió identificar dos aspectos claves que debería tener el modulo FIS:

- Interfaz de diseño en la cual se puedan configurar todas las características del FIS, como operadores lógico, método de agregación, método de desborrosificacion, variables lingüísticas de entrada y de salida, y la base de reglas.

- Una interfaz para la prueba del FIS, en la cual se pueda observar las salidas producidas por el FIS para determinadas entradas.

Como resultado del estudio de la literatura y del software existente de Lógica Difusa, se decidió, que el modulo FIS, permitiera implementar solo Sistemas de Inferencia Difusa (FIS) tipo Mandani por una razón:

[22] UNFUZZY 1.2: Herramienta de Software para el Análisis, Diseño, Simulación e Implementación de Sistemas de Lógica Difusa". Universidad Nacional de Colombia, sede Bogotá (1998). Información disponible en: http://www.ing.unal.edu.co/~electrica/Proyectos/Proyecto5.html

1. Este un proyecto cuyo objetivo central es el desarrollo de software y no el estudio en profundidad de la Lógica Difusa, por estar el sistema tipo Mandani ampliamente documentado se elige.

Aunque queda abierta la posibilidad de implementar otros tipos de sistemas en desarrollos futuros.

Otros aspectos de Lógica Difusa que se tomó la decisión de no implementar son:

- ✓ Modificadores Lingüísticos ver apartado 2.2.6.
- ✓ Para borrosificar se eligió el método **Max-Min** ver apartado 2.2.11**.**

4. FASE DE DESARROLLO

4.1 INTRODUCCIÓN

Como se afirmó anteriormente para la fase de desarrollo del proyecto, se sigue la metodología de construcción por prototipos y modelado orientada a objetos.

Al ser el modulo FIS una pieza, que debía ser integrada al software EVOLUCION, se presentaron dos requerimientos básicos: primero, el modulo FIS debe funcionar bien sin ser parte de EVOLUCION y segundo, el modulo FIS debe funcionar bien siendo parte de EVOLUCION y este ultimo debe seguir funcionando correctamente y conservar las funcionalidades que tenia antes de la integrarle el modulo FIS. Para satisfacer estos requerimientos se plantearon tres prototipos así:

- ✓ **Primer Prototipo:** implementa todas las funcionalidades del FIS.
- ✓ **Segundo Prototipo:** corrige los errores del primero y se integra a EVOLUCION.
- ✓ **Tercer Prototipo:** corrige los errores del segundo prototipo.

En lo que resta del capitulo se explicaran en detalle cada uno estos prototipos. Para la especificación de requisitos se siguen las directrices dadas por el estándar ANSI/IEEE 830[23].

Nota: para las pruebas se eligió matlab por ser un producto de amplia aceptación mundial y reconocimiento por su calidad.

[23] IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE 830. 1998

4.2 PRIMER PROTOTIPO

El primer prototipo implementa todas las funcionalidades básicas de un FIS. para definirlas se utilizaron, los conocimientos adquiridos durante la **FASE DE FORMACIÓN.** en este apartado se documentan los procesos de **análisis, diseño, desarrollo.**

4.2.1 Análisis. En esta fase se definen los requisitos del modulo FIS.

Especificación de requisitos:

Propósito: definir cuales deben ser los requerimientos que debe cumplir el modulo FIS.

Ámbito: el producto final será un software que permita diseñar fis, este tiene que poder ser usados para desarrollos futuros que involucren lógica difusa. la precisión de los resultados entregados por el componente deben estar en el orden 1%.

Definiciones, acrónimos y abreviaturas.

Tabla 1. Definiciones

Variable Lingüística	Son las variables sobre las cuales operan los FIS, esta compuesta por conjunto borrosos y las variables de entrada tienen asociado un difusor.
Conjunto Borroso	Representa los conjuntos en los cuales se dividen las variables lingüísticas. el grado de pertenencia de un objeto al conjunto se calcula Con la función de menbrecia o pertenencia, asociada al mismo.
Función de membresía.	Se utiliza para definir el grado de pertenencia de elemento a un conjunto borroso.

Dominio	Define el rango de valores que puede tomar una variable. Es aplicable a **variables lingüísticas, conjuntos borrosos, difusores y funciones de membresía.**
Borrosificador	Se encarga de convertir los valores de entrada en valores difusos, con los cuales opera el FIS.
Difusor	Conjunto que se asocia a la entrada, con el fin de eliminar el ruido. Es utilizado por el **Borrosificador.**
Deborrosificador	se encarga de transformar los valores difusos obtenidos al terminar las evaluaciones de las reglas y convertirlos en valores dentro del dominio de las variables de salida
Base de Reglas	Contiene las reglas con las cuales se toman las decisiones, en concordancia a las entradas.
Unión	Representa la unión entre conjuntos borrosos, definido por una t-conorma, en lógica difusa representa el operador **"OR"** ó **"O"**.
Intersección	Representa la intersección entre conjuntos borrosos, definido por una t-norma, en lógica difusa representa el operador **"AND"** ó **"Y"**.
Implicación	Operador lógico que es utilizado para determinar el valor de activación de la regla, en lógica difusa representa el operador **"THEN"** ó **"ENTONCES"**.
Complemento	Representa la en que grado un elemento no pertenece a un **conjunto borroso.**
Agregación	Función que determinan la forma como se combinan las reglas activadas por una entrada particular, para determinar un conjunto de salida.
Regla	Forma como se representa el conocimiento[24] tiene las formas "**SI *condición AND condición* ENTONCES X ES b",** "**SI *condición OR condición* ENTONCES X ES b".**
Condición	Tiene la formas "**Variable Lingüística ES Conjunto Borroso", "*NOT* Variable Lingüística ES Conjunto Borroso".**

[24] PRINCIPIOS DE INTELIGENCIA ARTIFICIAL & SISTEMAS EXPERTOS, Martha Vitalia Corredor Montagut. Pag 41.

Tabla 2 Abreviaturas

FIS	Sistema de Inferencia Difusa. Consta de un **Borrosificador, una Base de Reglas, y un Deborrosificador.**

Descripción general. El modulo FIS, es solo uno, de los tantos esfuerzos realizados por el grupo SIMON para integrar la Dinámica de sistemas y Lógica Difusa. Este modulo implementa las funcionalidades de un FIS tipo Mandami.

El modulo debe proporcionar todas las herramientas necesarias para la creación de FIS. En este apartado se presentan todas las funciones que debe realizar, restricciones, entradas y salidas.

Funciones del modulo FIS. A continuación se detallan las funciones que debe realizar. Se puede decir que el modulo solo realiza dos tareas en concreto:

1. **Crear Sistemas de Inferencia Basados En Lógica Difusa.**
2. **Inferir un conjunto de valores de salida, para un conjunto de valores de entradas.**

Para cumplir la primera tarea, se hace necesario realizar una serie de subtareas:

Agregar variables lingüísticas de entrada: implica la creación de una variable lingüística, la cual esta compuesta por:

1. Uno o más conjuntos borrosos.

2. Un difusor.
3. Un dominio.
4. Un nombre.

Luego que la variable esta definida, es agregada al FIS.

- Agregar variables lingüísticas de salida: es igual a la función anterior, pero no se le define difusor.

- Eliminar variables lingüísticas: Borra la variable lingüística del FIS.

- Modificar variables lingüísticas: Cambia cualquiera de las características de la variable lingüística.

- Agregar conjunto borroso: se crea un conjunto, el cual será agregado a una variable lingüística, todos los conjuntos presentan las siguientes características.

1. Un nombre.
2. Función de membresía.
3. Un dominio que es igual, al de la variable lingüística, a la cual será agregado el conjunto.

- Modificar conjunto borroso: cambia cualquiera de las características del conjunto.

- Eliminar conjunto borroso: elimina un conjunto de la variable lingüística.

- Establecer difusor: se crea para ser asignado a una variable lingüística de entrada, un difusor esta caracterizado por:

1. Un nombre.
2. Función de membresía.
3. Un dominio.
4. Uno o más puntos de evaluación.

- Modificar difusor: cambia cualquiera de las características del difusor.

- Crear base de reglas.

- Crear regla: una regla esta determina por las siguientes características:

1. Definición.
2. Peso.

- Definir peso de la regla: le asigna un grado de importancia para regla. Este es un número entre 0 y 1.

- Modificar peso de la regla: cambia el peso que ha signado a la regla.

- Agregar regla a la base de reglas: adhiere una nueva regla a la base de reglas.

- Eliminar una regla, de la base de reglas: retira la regla de la base de reglas.

- Modificar una regla, de la base de reglas: consiste en cambiar la definición o peso de una regla que ya fue agregada a la base de reglas.

- Crear los operadores para la unión, intersección, complemento, implicación, agregación: consiste en asignar la función que define a cada uno de los operadores.

- Cambiar operadores: se refiere a modificar los operadores que definen la unión, intersección, complemento, implicación, agregación: consiste en cambiar la función que define a cada operador.

Para cumplir con la segunda tarea se hace necesario, realizar las siguientes subtareas:

- Borrosificar las entradas: se toma la entrada y se determina a que variable lingüística pertenece, luego se calcula a que grado pertenece a cada uno de los conjuntos borrosos de la variable lingüística.

- Identificar las reglas, que son activadas por los valores de entradas.

- Identificar que conjuntos borrosos y de cuales variables lingüistas de salida son afectados por las reglas activadas.

- Realizar el proceso de **Deborrosificación,** para determinar el valor de salida**:** para esta tarea se utilizan los métodos de deborrosificación presentados en el capitulo 2.

Características de los usuarios. El modulo FIS tiene dos clases de usuarios plenamente identificados:

- Desarrolladores que agregarán las el FIS o clases de este, para sus propios desarrollos.

- Usuarios que desean implementar un FIS para la solución o el estudio de un problema.

Restricciones generales. Al tener el modulo FIS dos clases de usuarios, se crea la necesidad de plantear restricciones para cada uno de ellos.

Los desarrolladores deben conocer sobre UML, programación orientada a objetos, el lenguaje de programación DELPHI y algunos conocimientos básicos de Lógica Difusa.

En cuanto a la otra clase de usuarios, se requiere que estos estén familiarizados con el ambiente de trabajo basado en ventanas de Windows, conocimientos básicos de Lógica Difusa.

Casos de uso. En la figura 10 se puede ver el diagrama de casos de uso del primer prototipo.

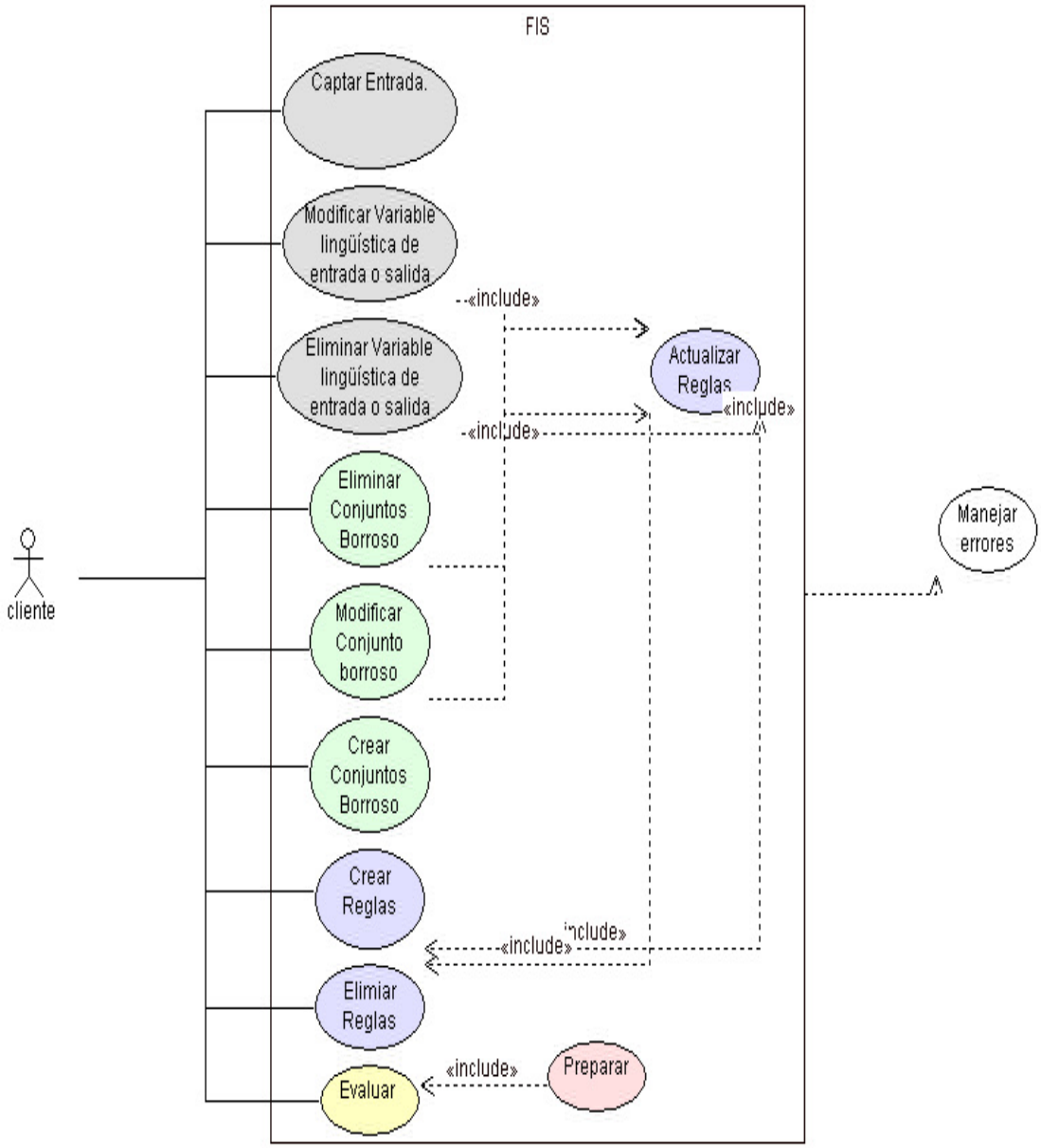
- El actor Cliente representa cualquier entidad que utilice el modulo FIS, puede ser un modelador, otro software, un desarrollador que este utilizando el software para resolver un problema especifico.

Los casos de uso presentados a continuación, le corresponden al actor Cliente.

- **Captar entrada:** el FIS recibe las entradas que le son suministradas por el Cliente.

- **Modificar variable lingüística de entrada o salida:** le permite al cliente cambiar cualquiera de las características de la variable lingüística.

Figura 10. Diagrama de casos de uso. Primer prototipo.



▪ **Eliminar variable lingüística de entrada o salida:** el cliente tiene la posibilidad de retirar del FIS la variable lingüística que considere, ya se de salida o entrada.

- **Eliminar conjunto borroso:** este caso de uso esta asociado al caso de uso **modificar variable lingüística de entrada o salida,** permite eliminar un conjunto borroso de la variable lingüística.

- **Modificar conjunto borroso:** este caso de uso esta asociado al caso de uso **modificar variable lingüística de entrada o salida,** permite modificar cualquiera de las características de un conjunto borroso, que pertenece a una variable lingüística.

- **Crear conjunto borroso:** le permite al Cliente crear un conjunto borroso, definiéndole todas las características del mismo.

- **Crear regla:** el Cliente define la regla de inferencia determinando sus características y la incluye a la base de reglas del FIS.

- **Eliminar regla:** le permite al Cliente retirar una regla, de la base de reglas del FIS.

- **Evaluar:** le permite al Cliente determinar, el valor o valores de salida para un conjunto de entrada.

El caso de uso que se presenta a continuación es implícito a otros casos de uso como se muestra en la Figura 10.

- **Actualizar reglas:** se actualizan las definiciones de las reglas de las base de reglas.

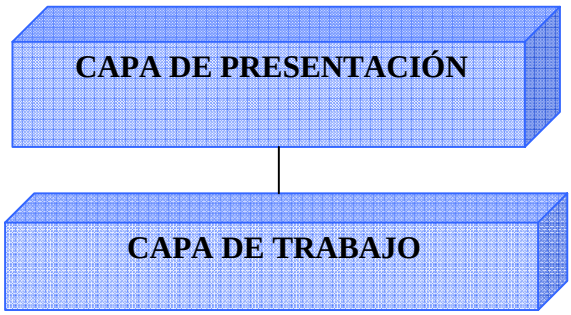
4.2.2 Diseño de componentes. Producto del proceso de análisis, se logró estructurar un modelo para el modulo FIS el cual satisface todos los requerimientos.

Este apartado esta dedicado a explicar los componente mas importantes, que hacen parte del modulo FIS.

Nota: el nombre de las clases comienza con la letra mayúscula T y el de las interfaces con la letra mayúscula I.

Diseño Arquitectónico. "El diseño de software orientado a objetos requiere la definición de una arquitectura de software multicapa, la especificación de subsistemas que realizan funciones necesarias y proveen soporte de infraestructura, una descripción de objetos, que son los bloques de construcción del sistema y una descripción de los mecanismos de comunicación, que permitan que los datos fluyan entre las capas, subsistemas y objetos. El diseño orientado a objetos cumple con todos estos requisitos"25. Para el desarrollo del componente FIS, se utilizó una arquitectura de dos capas, una capa de trabajo que se encarga del procesamiento de realizar todas las tareas del FIS y otra capa para presentar y captar información de trabajo. La Figura 11, muestra esquemáticamente esta arquitectura.

Figura 11. Diseño Arquitectónico del componente FIS.

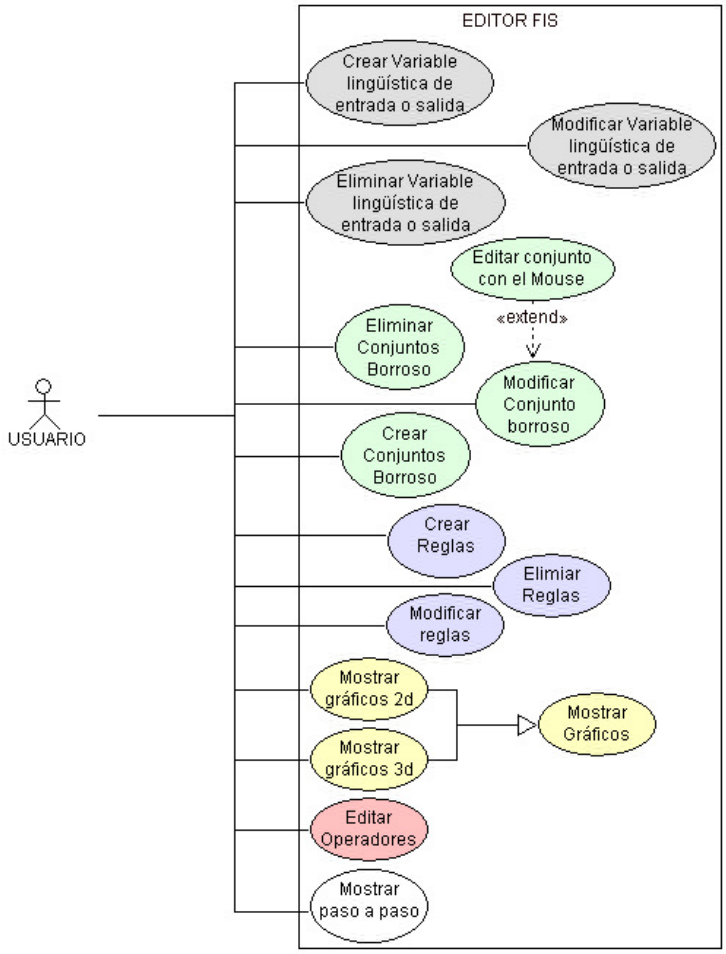


[25] PRESSMAN, Diseño Orientado a Objetos. Op. cit, p. 379.

- **Capa de Presentación:** básicamente esta capa es la encargada de la presentación de información al usuario, a través de esta el usuario interactúa con el componente FIS. Esta representada por el editor FIS. La figura 12, muestra el diagrama de casos de uso Editor FIS.

- **Capa de Trabajo:** se encarga de proporcionar las funcionalidades del software, como son lo es el proceso de Inferencia. El diagrama de casos de uso es mostrado en la figura 10.

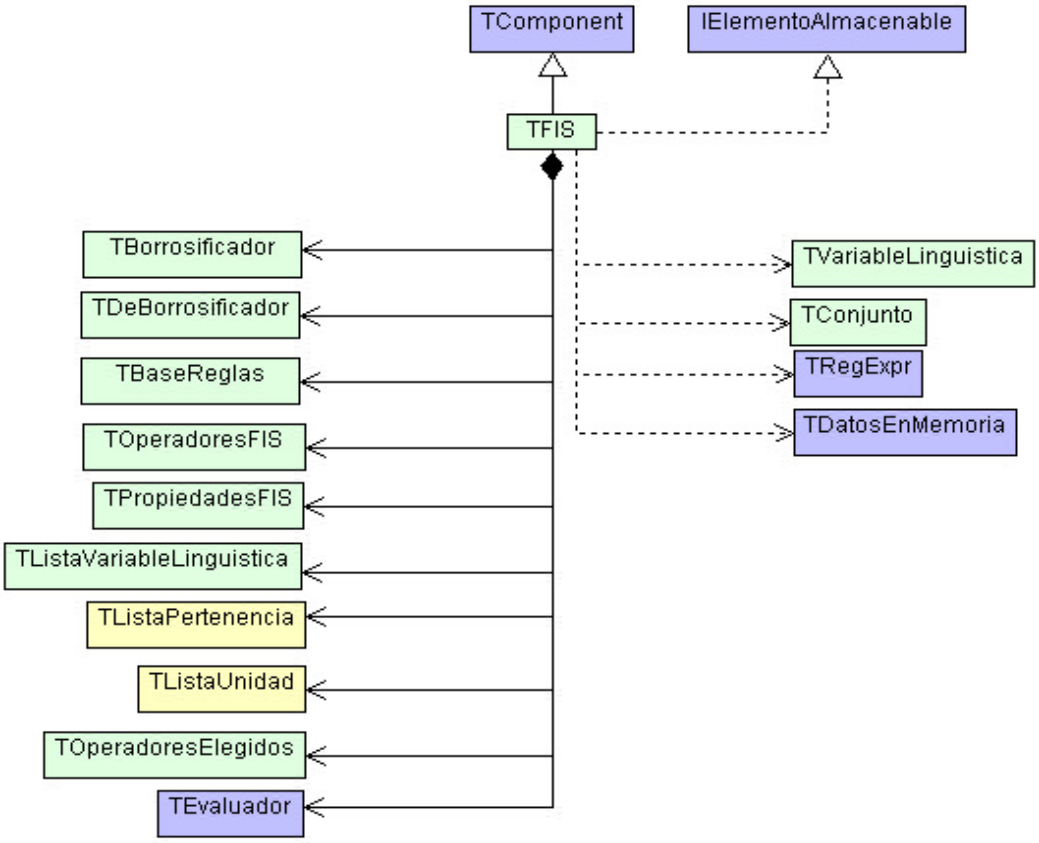
Figura 12. Diagrama de casos de uso, Editor FIS.



4.2.3 Estructura de clases. Las clases del modelo, se dividieron en tres submodelos así:

TFIS: es la clase principal, representa el FIS como tal, proporciona la funcionalidad principal del componente “Evaluar entradas”. La figura 13, muestra su diagrama de clase.

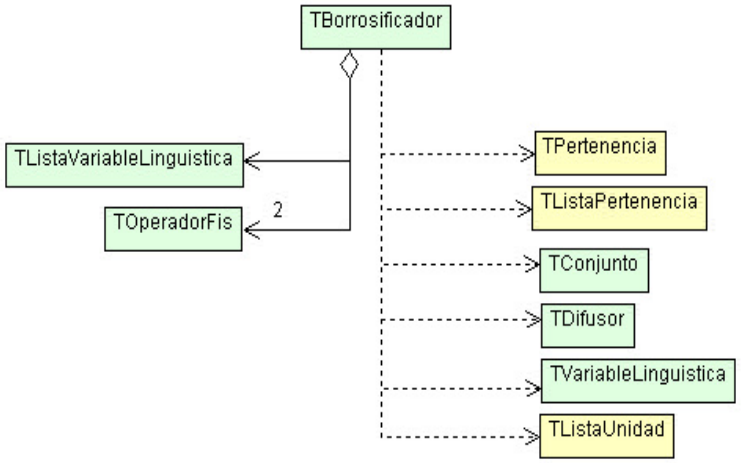
Figura 13. Diagrama de clases TFIS.



TBorrosificador: Se encarga de convertir las entradas no borrosas en valores borrosos. Recibe como entrada un objeto **TListaUnidad** y entrega como salida un objeto **TListaPertenencias,** estos objetos serán explicados

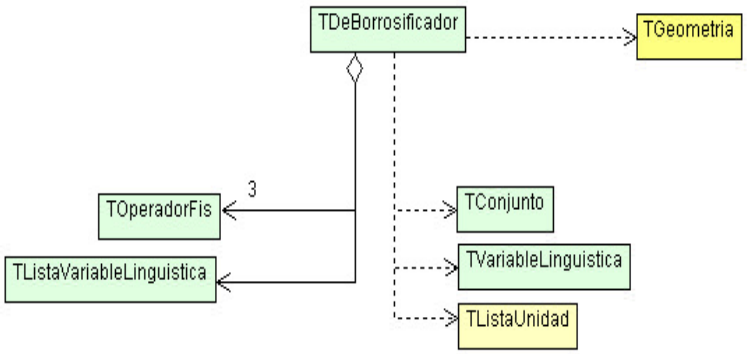
en detalle más adelante. La figura 14, muestra el diagrama de clases de **TBorrosificador.**

Figura 14. Diagrama de Clases de TBorrosificador.



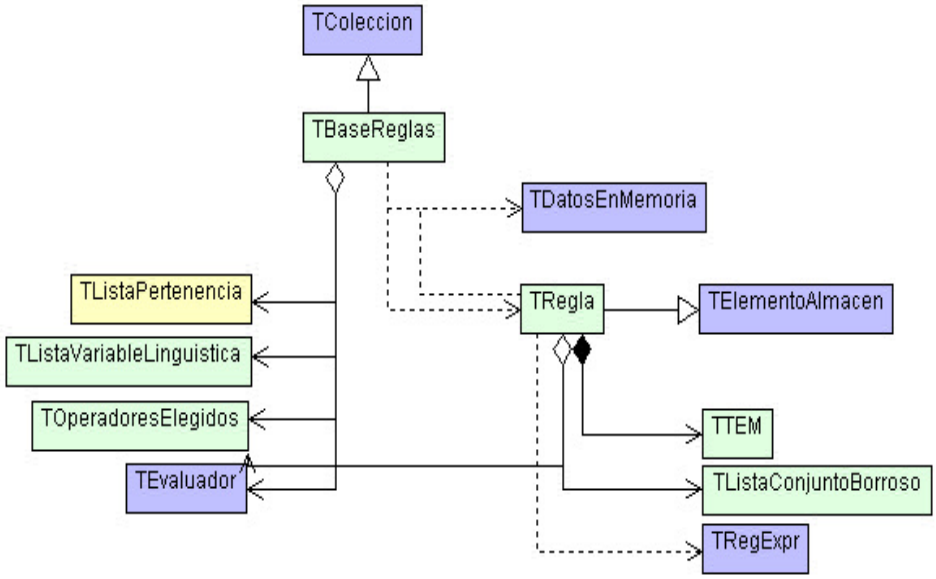
TDeBorrosificador: Se encarga de tomar los valores de las reglas y los convierte en valores dentro del dominio de las variables lingüísticas de salida. Recibe como entrada un objeto **TListaVariableLinguistica** y entrega como salida un objeto **TListaUnidad.** La figura 15, muestra el diagrama de clases del **TDeBorrosificador.**

Figura 15. Diagrama de Clases TDeBorrosificador.



TBaseReglas: Se encarga de gestionar las reglas, así como la activación de las mismas, como respuesta a una entrada. Recibe un objeto **TListaPertenencias** y activas las reglas pertinentes. La grafica 16, muestra el diagrama de clases de **TBaseReglas.**

Figura 16. Diagrama de Clases TBaseReglas.



Las clase comunes a todas las clases o cuyo propósito es prestar servicios que le puedan servir tanto al FIS como a otros desarrollos se agruparon en un diagrama cuyo nombre es Diagrama de Clases comunes. A continuación se detallan las clases más importantes.

TRegExpr: se utiliza para trabajar con expresiones regulares[26], su licencia es freware, fue desarrollada por **Andrey V. Sorokin,** Saint Petersburg, Russia, anso@mail.ru , disponible en http://RegExpStudio.com.

[26] Expresiones Regulares

TEvaluador: permite evaluar expresiones matemáticas ver manual del programador de EVOLUCION.

TVariableLinguistica: modela una variable lingüística como se define en el ámbito de la Lógica Difusa.

TConjuntoBorroso: modela un conjunto borroso como se define en la lógica Difusa.

TDifusor: Se utiliza para con ayuda de la función de membresía de los conjuntos borrosos calcular el grado de pertenencia de un elemento a los conjuntos.

TUnidad: es el medio con el cual el FIS se comunica con el exterior y también es utilizada en algunas comunicaciones internas. Consta de, primero un valor (Extended) y nombre Variable Linguistica (String).

TlistaPertenencias: Se utiliza principalmente para la comunicación interna en el FIS. Consta de nombreVariableLinguistica (String), nombre Conjunto Borroso (String), gradoPertenencia (Extended).

TDominio: modela el dominio como se le conoce en matemáticas. Consta de limiteInferior (Extended), limiteSuperior (Extended). Se utiliza para definir el dominio de las **funciones de membresía, conjuntos borrosos y variables lingüísticas.**

FiguraGeometrica: se utiliza para almacenar características de los conjuntos borrosos como centro de masa y área.

TDatosEnMemoria: Esta clase es usada para guardar archivos en memoria y luego pasarlos a un archivo físico o viceversa. Ver manual del programador de EVOLUCION.

TOperador: se utiliza para representar un operador cualquiera, definido como una función de varias variables.

TOperadorFIS: modela los operadores lógicos (AND, OR, IMPLICACIÓN) utilizados por FIS.

TElementoAlmacen: clase base de cualquier tipo elemento que se guarde en un archivo en EVOLUCIÓN. Ver manual del programador de EVOLUCIÓN.

TFuncion: esta clase representa una función de una sola variable independiente es utilizada para componer las funciones de membresía.

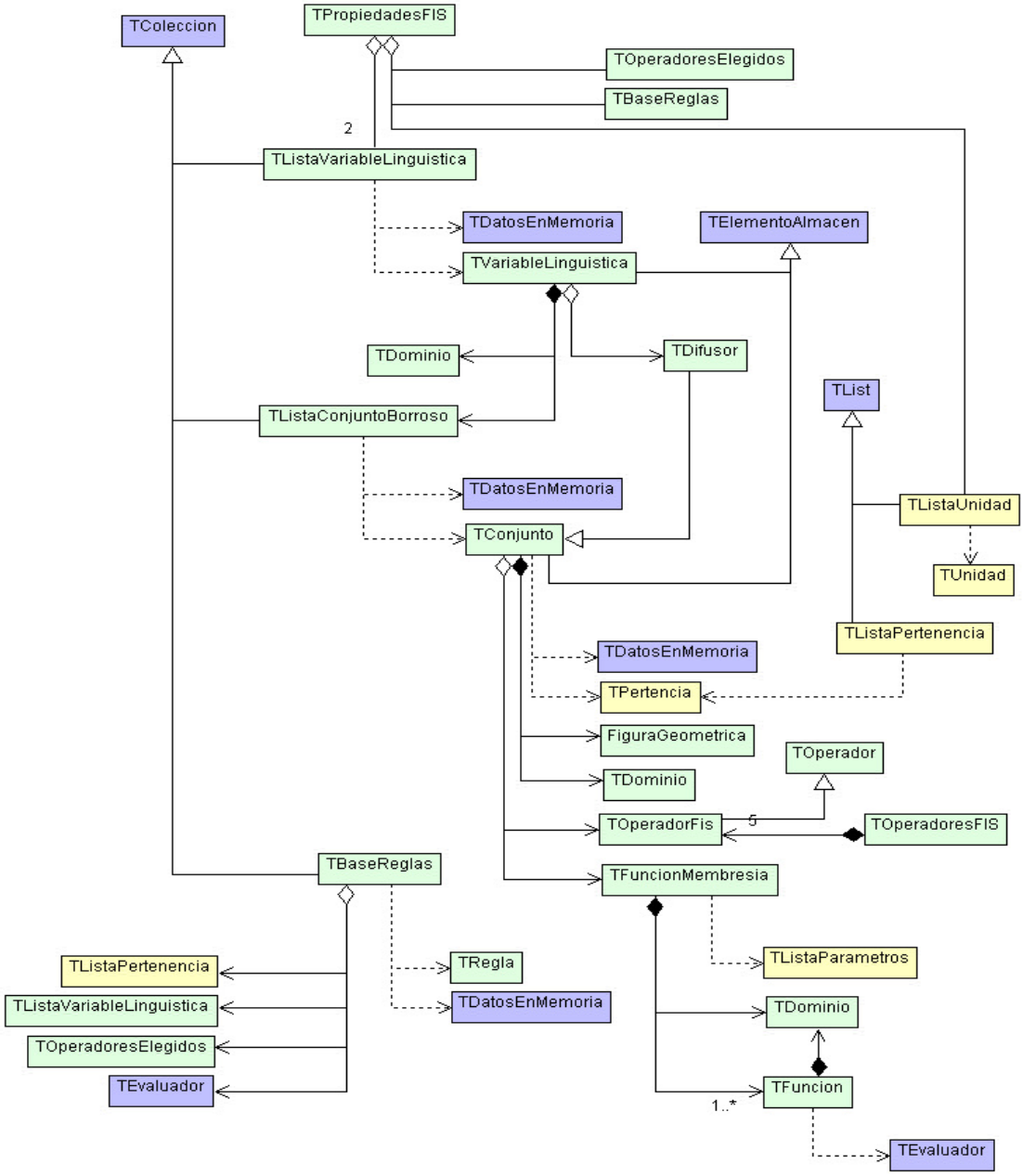
TFuncionMembrecia: Modela una función de membresía como se le conoce en el campo de la Lógica Difusa.

TGeometria: se utiliza para calcular las caracteristicas geometricas del conjunto borroso.

TTEM: transforma expresiones matemáticas en expresiones que contengan los operadores de la lógica borrosa, tales como “a*b“ a la expresión “min(a,b)”

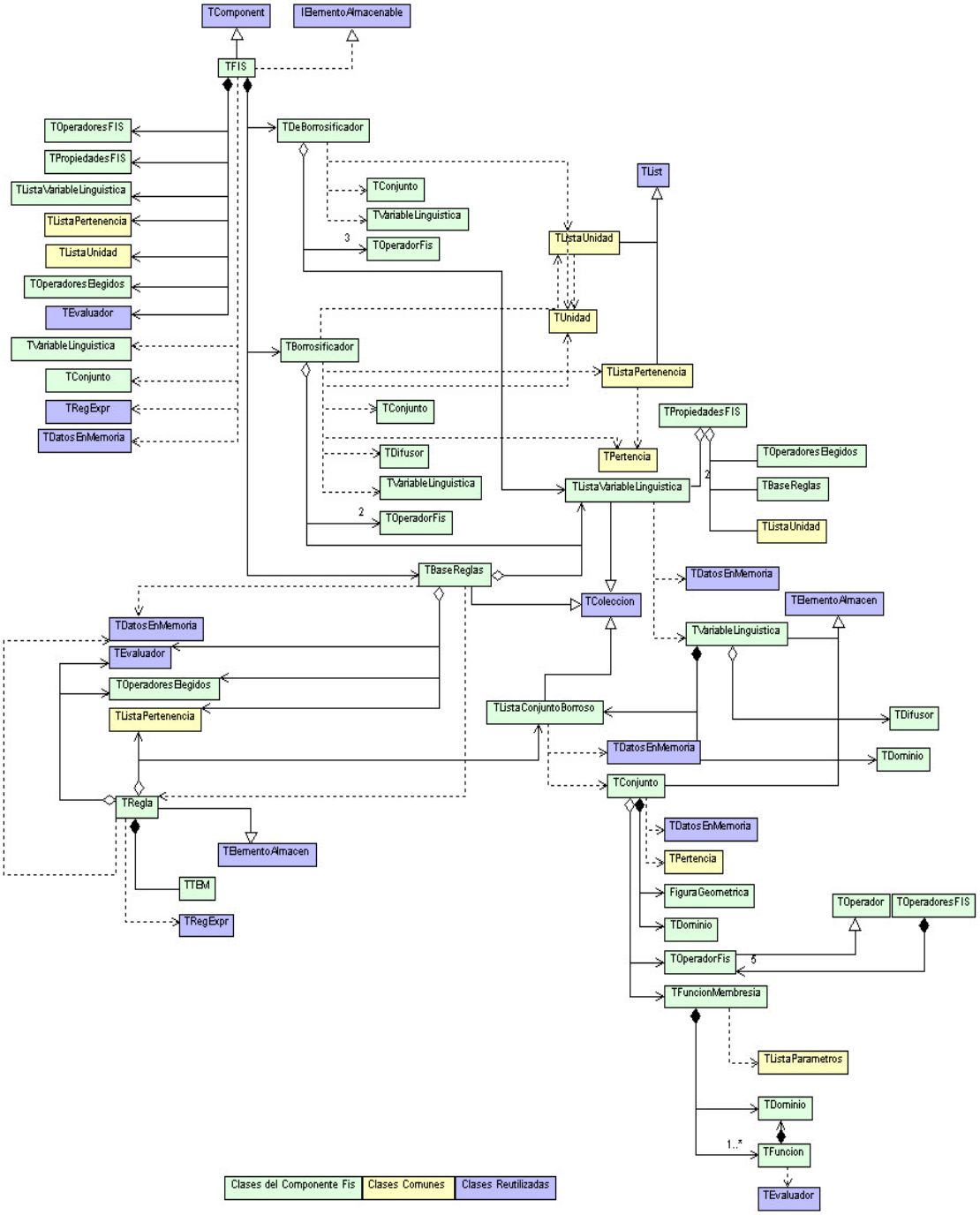
La figura 17, muestra el diagrama de clases comunes.

Figura 17. Diagrama de Clases Comunes.



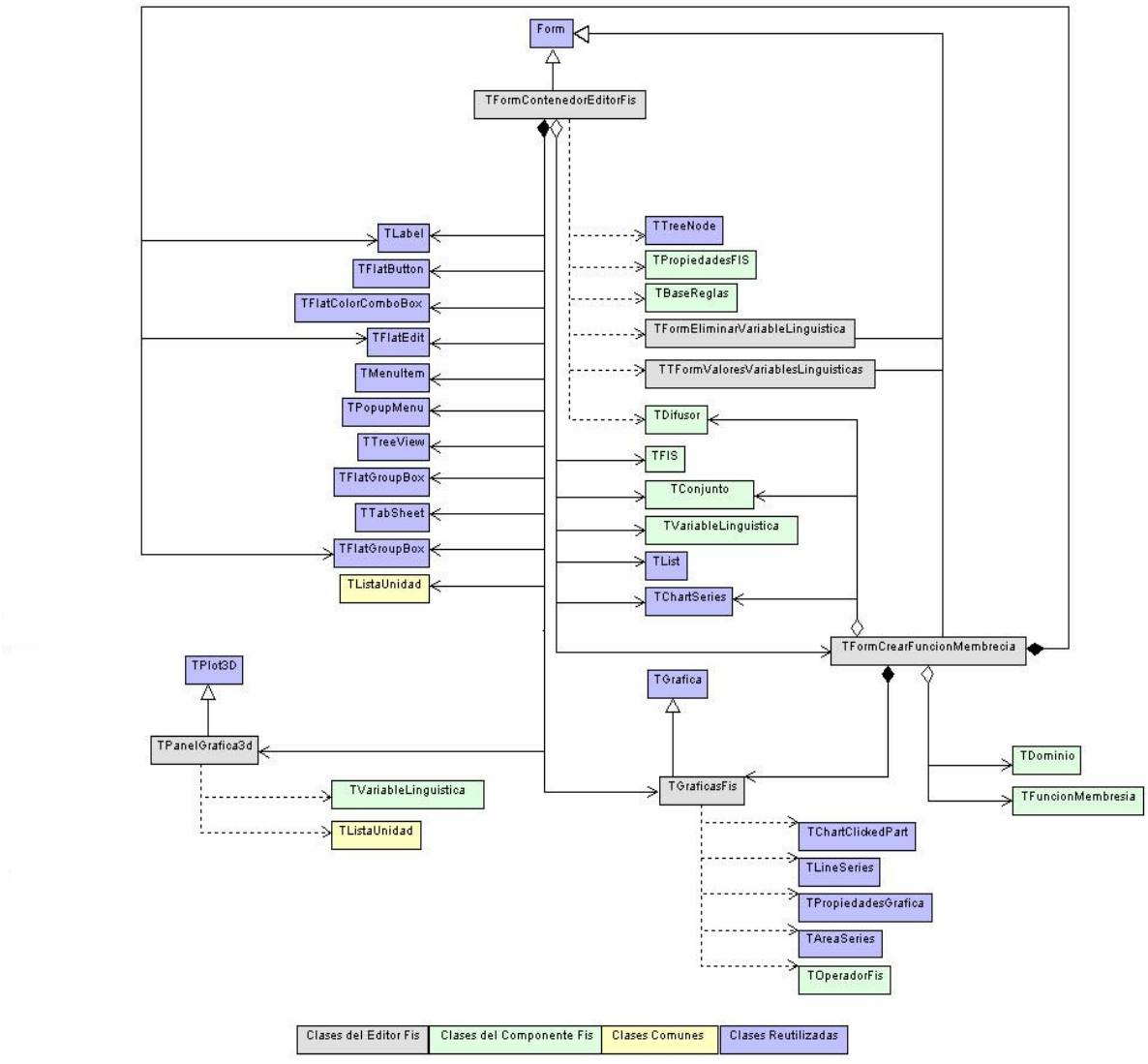
Ya se presentaron cada una de las clases del componente FIS, en la figura 18, se muestra el diagrama de clases completo del componente.

Figura 18. Diagrama de Clases Componente FIS.



Hasta el momento se presentó el diseño del corazón del modulo FIS, ahora se presentará el diseño de la interfaz grafica del modulo.

Figura 19. Diagrama de Clase Editor FIS.



TFormContenedorEditorFIS: es la clase principal de la interfaz grafica del componente FIS. Se encarga de manejar la interacción con el usuario. La figura 17, muestra su diagrama de clases.

A continuación se describen las principales clases de este componente:

- **TPanelGrafica3D:** Se utiliza para graficar en 3 dimensiones, las variables lingüísticas de salida.

- **TPlot3D:** Clase para graficar en 3 dimensiones, se obtuvo bajo licencia Freware. Disponible en la dirección Web www.lohninger.com/.

En el anexo A se presentan los diagramas de secuencias de las tres funciones principales del componente FIS.

- **Iniciar:** en esta tarea se crean todos los objetos que necesita el FIS para operar.

- **Preparar:** realiza todos los procesos necesarios para que el componente FIS inicie su operación.

Evaluar: si el FIS esta preparado evalúa, si no lo esta, se prepara y luego evalúa.

4.2.4 Implementación. Al concluir la fase de implementación el componente FIS cuenta con los requerimientos mínimos necesarios para creación de **Sistemas de Inferencia Difusa tipo Mandani,** tanto a nivel de capa de presentación como de capa de trabajo.

Estas funciones son:

- ✓ **Agregar variable lingüísticas.**
- ✓ **Eliminar variables lingüísticas.**

- ✓ **Definir los operadores lógicos (AND, OR, IMPLICACIÓN, COMPLEMENTO).**
- ✓ **Definir método de desborrosificación.**
- ✓ **Definir método de agregación.**
- ✓ **Agregar reglas.**
- ✓ **Eliminar reglas.**
- ✓ **Captar datos de entrada.**
- ✓ **Proporcionar datos de salida.**
- ✓ **Evaluar.**

4.2.5 Pruebas. Como el objetivo principal del componente FIS, es integrarse a evolucion, se tomo la decisión de no realizarle pruebas rigurosas al primer prototipo. esto con el fin de avanzar rápidamente en el proceso de integración, una vez realizada la integración, se procede a realizar pruebas exhaustivas del funcionamiento del componente FIS, en tres aspectos.

- **Calidad de los resultados que entrega:** esto se verifica comparando sus resultados con los ofrecidos por otro software, para esto se utilizará la toolbox de Lógica Difusa de MatLab.

- **Pruebas de integración con EVOLUCION:** se realizarán modelos que implemente FIS, se probaran modelos que no utilicen FIS para comprobar que conserva todas sus características.

- **Interacción de los usuarios con la interfaz gráfica del FIS:** se realizarán seminarios de Dinámica de Sistemas y Lógica Difusa con la ayuda de la herramienta para lograr que los usuarios utilicen la herramienta y lograr detectar errores.

4.3 SEGUNDO PROTOTIPO.

Como se mencionó el objetivo de este prototipo es lograr la integración con EVOLUCION del componente FIS.

4.3.1 Análisis. Para el segundo prototipo, el componente FIS, conserva todas las funciones definidas para el primer prototipo y no agrega nuevas debido a que su objetivo es la integración con evolución.

Integrar el componente FIS a EVOLUCIÓN exige el cumplimiento de unas especificaciones de diseño como son:

- Implementar una serie de interfaces: por ejemplo. **IPropiedades Graficas, IObjeto Pintable.**

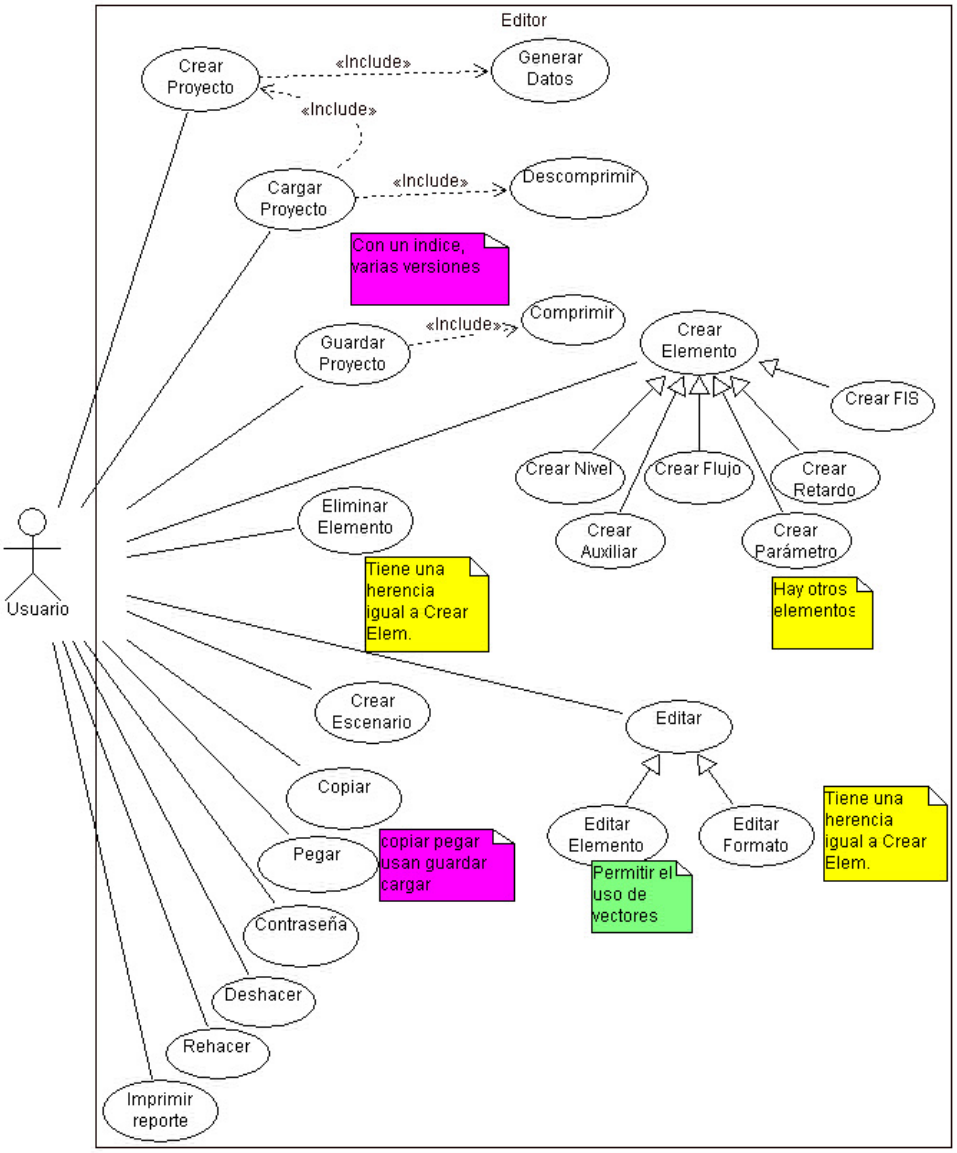
- Heredar de unas clases: por ejemplo, **TElemento, TElemento EValuación.**

- Crear unas clases con unas condiciones de herencia claramente definidas.

Agregar este elemento generó un cambio en el diagrama de casos de uso del Editor[27] de EVOLUCION. El cambio consiste en un nuevo caso de uso llamado **Crear FIS** el cual permite crear un FIS en el diagrama Forrester. La Figura 20, muestra el nuevo diagrama de casos de uso del Editor.

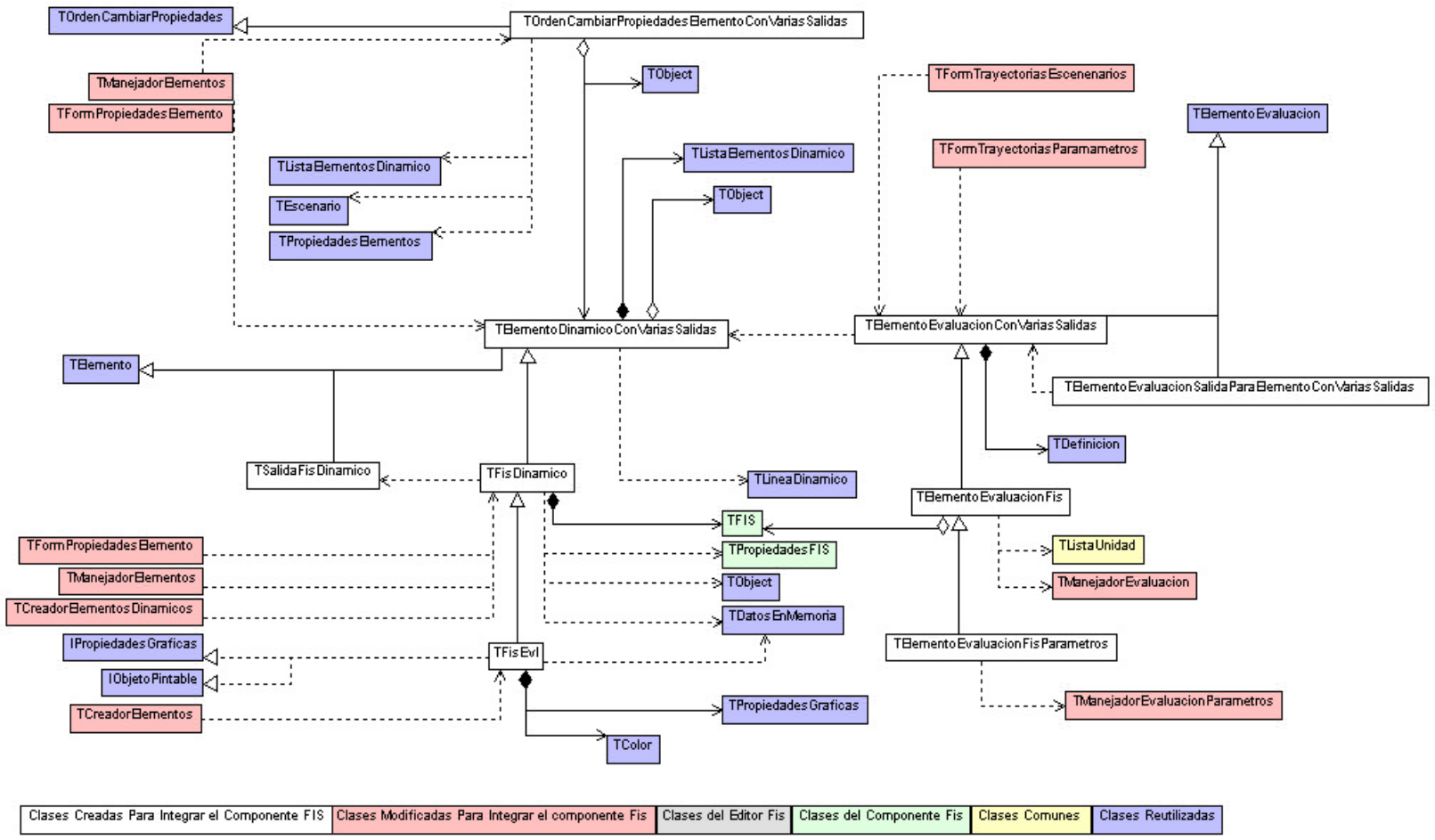
[27] Editor: Provee los mecanismos para la creación, edición y destrucción de elementos del diagrama Forrester.

Figura 20. Diagrama de Casos de uso Editor EVOLUCION.



4.3.2 Diseño. Luego de estudiar a fondo el diseño de evolución se logró integrar el modulo FIS, como se muestra en el diagrama de clases de la figura 21.

Figura 21. Diagrama de clases del componente FIS, integrado a EVOLUCIÓN



El principal problema que se encontró para la integración del componente FIS a EVOLUCION, fue la imposibilidad de este para gestionar elementos de varias salidas. Para solucionar este problema se planteo la estructura de clases siguiente:

- **TElementoDinamicoConVariasSalidas:** Es una clase de tipo TElemento que además permite trabajar con varias salidas. Las salidas son objetos del tipo TElemento.

- **TSalidaFisDinamico:** Es un clase de tipo TElemento que representa una salida para el elemento dinámico FIS.

- **TFisDinamico:** Es un elemento dinámico con varias salidas que contiene un TFIS.

- **TFisEvl:** es la clase que permite que evolución pinte en el diagrama de Forrester un elemento FIS.

- **TElementoEvaluacionConVariasSalidas:** Es una clase utilizada para implementar la evaluación del elemento de varias salidas.

- **TElementoEvaluacionFis:** es la clase que controla la evaluación del componente FIS.

- **TElementoEvaluacionFisParametros:** es la clase que se crea para la interacción del elemento FIS en el análisis de sensibilidad por parámetros.

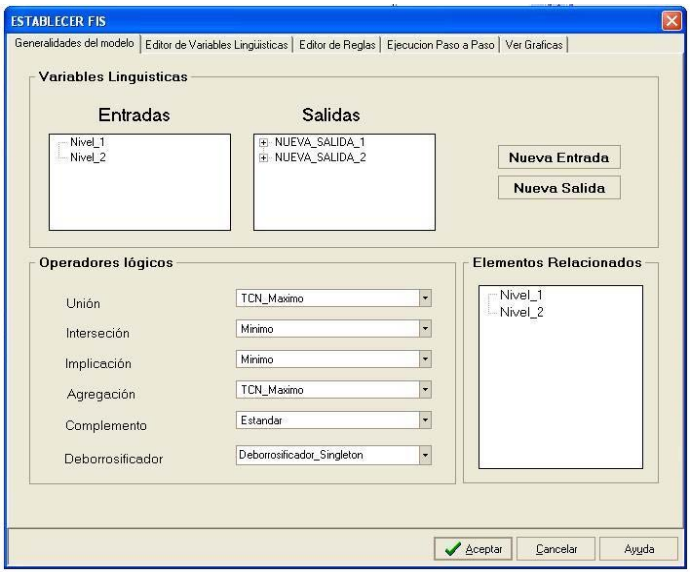
- **TElementoEvaluacionSalidaParaElementoConVariasSalidas:** es la clase hecha para que represente una salida del elemento con varias salidas.

▪ **TOrdenCambiarPropiedadesElementoConVariasSalidas:** permite el manejo del cambio en el elemento FIS, para la aplicación de la utilidad deshacer-rehacer. Esta estructura puede observarse en las clases de color blanco de la figura 21.

4.3.3 Implementación. Al terminar esta fase el componente FIS, ha sido integrado a EVOLUCION, permitiéndole la creación en tiempo de modelado la creación de FIS. Las funcionalidades que implementa son:

Definir generalidades de modelo: le permite al usuario definir las funciones principales del modelo. Como los operadores lógicos (OR, AND, IMPLICACIÓN y COMPLEMENTO), los métodos de agregación y desborrosificación, por ultimo visualizar las entradas y salidas del FIS. La figura 22, muestra la interfaz grafica de esta función.

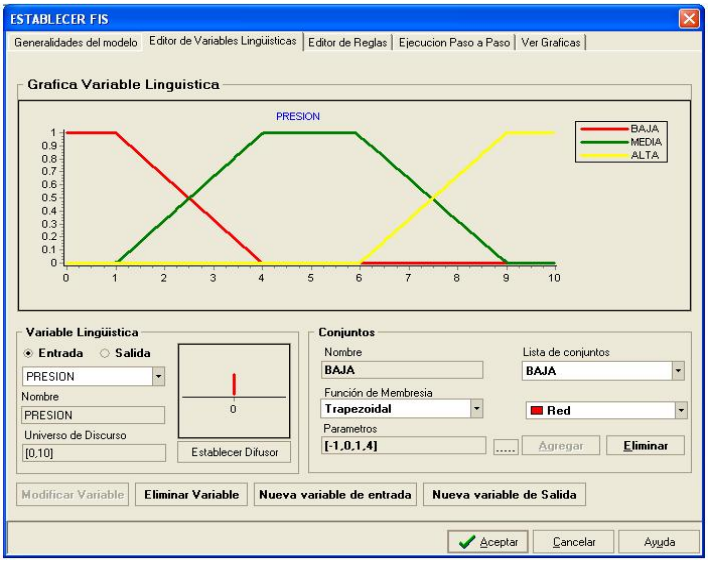
Figura 22. Generalidades del FIS.



Editor de variables lingüísticas: da la posibilidad de crear, modificar y eliminar las variables lingüísticas de entrada y salida del sistema. Además

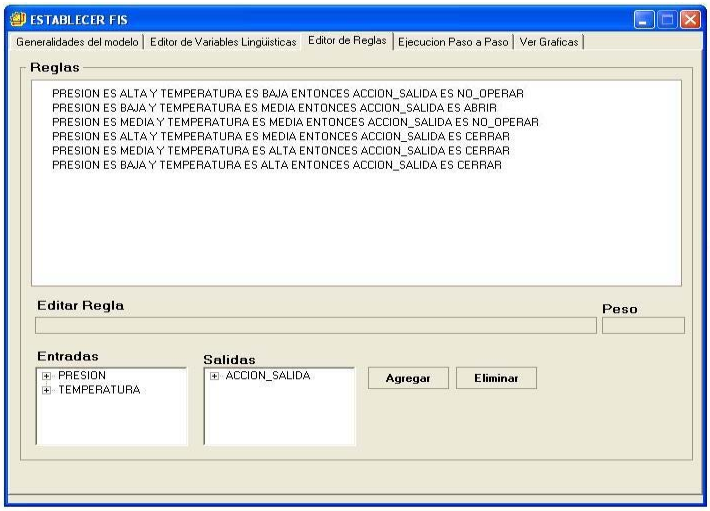
permite agregar, modificar y eliminar los conjuntos borrosos de las variables lingüísticas y establecer el difusor de las variables de entrada. La figura 23, muestra la interfaz grafica para realizar dichas tareas.

Figura 23. Editor de variables lingüísticas.



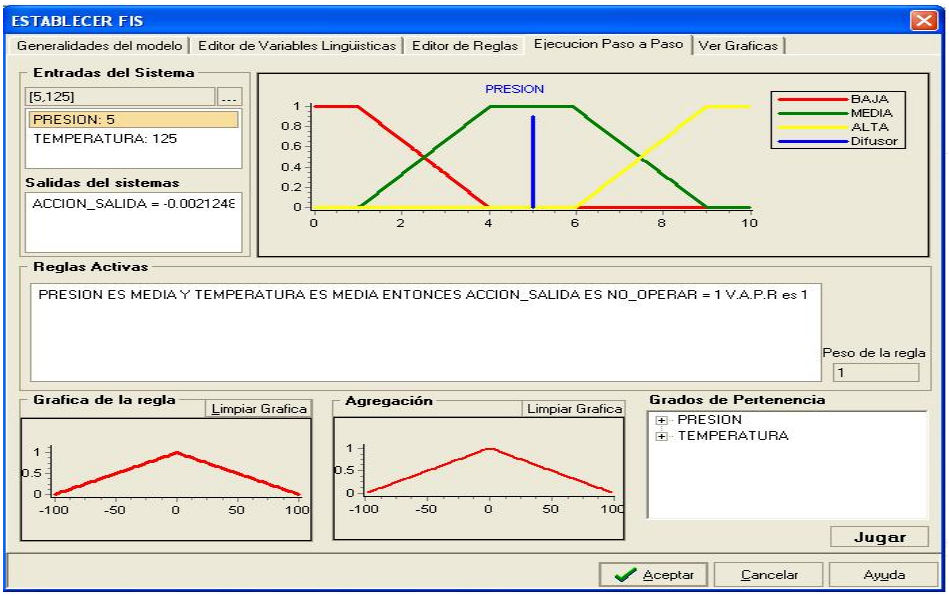
Editor de Reglas: permite crear, modificar y eliminar reglas de la base de reglas. La figura 24, muestra su interfaz grafica.

Figura 24. Editor de reglas.



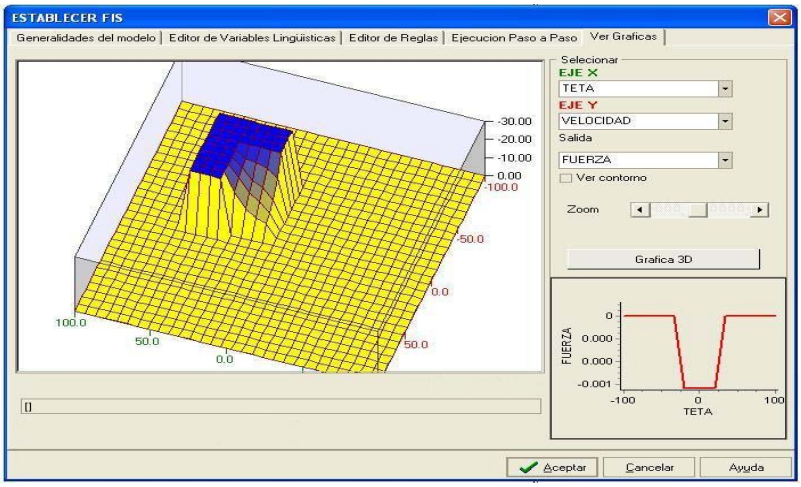
Ejecución paso a paso: permite probar las salidas del FIS, para un conjunto de entradas. La figura 25, muestra su interfaz grafica.

Figura 25. Ejecución paso a paso.



Ver grafica: permite observar la superficie de decisión del FIS. La figura 26 muestra su interfaz grafica.

Figura 26. Ver grafica.



4.3.4 Pruebas. Se realizó una sesión de pruebas con el director de proyecto. esta sesión tuvo como objetivos evaluar la interfaz grafica y el comportamiento de EVOLUCIÓN con el componente FIS. los resultados de esta fueron:

- ✓ La forma de ingresar las reglas era tediosa y muy susceptible a errores. Para solucionar esto se decidió que las reglas no deberían ser escritas por el usuario. Si no creadas con la ayuda de una asistente.
- ✓ Las graficas de los conjuntos borrosos deben permitir una edición grafica.
- ✓ Permitir crear variables lingüísticas de entrada o salida por defecto.
- ✓ El icono del FIS, no era visible en el entorno de trabajo de EVOLUCION.
- ✓ El componete FIS, no implenta, cortar, pegar, rehacer, deshacer.

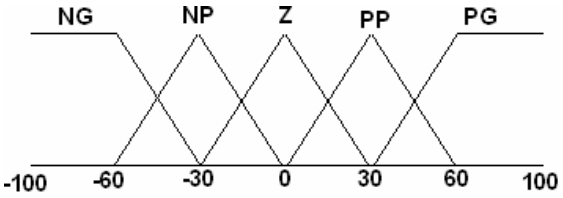
Para verificar que los cálculos del FIS, fuesen correctos se crearon una serie de FIS. Los cuales son presentados a continuación.
Se toma un ejemplo de sistema de inferencia borrosa, se hace la inferencia sobre algunos valores y se comparan con valores generados en otra herramienta (Matlab) y con los valores analíticos.

Ejemplo de FIS: tomado del libro: Bonifacio Martín del Brío, Alfredo Sanz Molina, Redes Neuronales y Sistemas Difusos, 2ª Edición, ALFAOMEGA RA-MA.

Péndulo Inverso: Se trata de mantener en posición vertical una varilla que se apoya en un carrito movido por un motor, sistema qu se asemeja a un péndulo invertido. El FIS fijará la fuerza F que debe realizar el motor del carrito; las entradas al FIS son el ángulo "TETA" que forma el péndulo con la vertical y su VELOCIDAD angular. En este caso, el vector de entradas al FIS es x=(TETA,VELOCIDAD), y la salida y=FUERZA. El rango de valores (Universo de discurso) para las entradas es [-100,100] para el ángulo, [-

100,100] para la velocidad angular; el rango de valores para la salida es [-100,100].

El siguiente paso en el diseño del controlador consiste en definir las particiones correspondientes a las variables a las variables Lingüísticas de entrada y salida. Para entradas y salidas se ha elegido crear particiones como se muestra a continuación.



Donde NG es Negativo grande, NP es negativo pequeño, Z es cero, PP es positivo pequeño, PG es Positivo grande. Ya se tienen las variables Lingüísticas con sus respectivas particiones borrosas, el siguiente paso es crear la base de reglas para el sistema, para el ejemplo se crean dos reglas así:

Regla_1: SI TETA ES Z Y VELOCIDAD ES Z ENTONCES F ES Z
Regla_2: SI TETA ES PP Y VELOCIDAD ES NP ENTONCES F ES NP

Regla_1 expresa que si el ángulo es cero y la velocidad angular también, entonces no hay que ejercer fuerza alguna (puesto que la varilla estará en equilibrio). La Regla_2, por su parte, indica que si el ángulo es pequeño positivo y la velocidad pequeña negativa, entonces la fuerza que se deberá ejercer debe ser pequeña negativa para tender al equilibrio. Desarrollada la base la base de reglas borrosas, se han de seleccionar después los métodos de borrosificación, de inferencia y de desborrosificación. Para esta elección se deben considerar fundamentalmente los aspectos relativos a la eficiencia computacional, para el ejemplo se elige el método de desborrosificación singleton, para las variables se elige un difusor singleton,

se elige el operador MAX para la unión y agregación, el MIN para intersección e implicación.

La tabla 3, muestra una comparación de lo valores de salida para el FIS, con 20 valores aleatorios de entradas. Dicha comparación se realiza entre los el valor analítico y los arrojados por el componente FIS y MatLab.

Tabla 3 Comparación de error FIS_Evolucion vs Matlab, prototipo 2

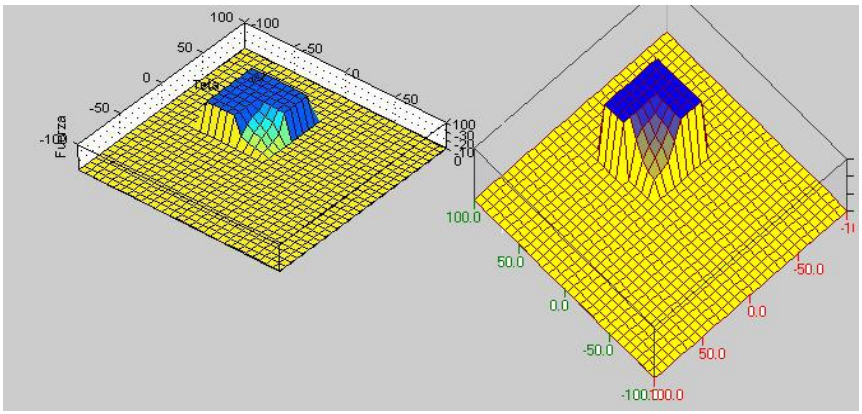
	Entradas		Salidas			error	
			FUERZA_FIS				
	TETA	VELOCIDAD	FIS_Evl	Matlab	Analitico	Fis-Analitico	Matlab-Analitico
1	-23,6869114	18,64345637	1,46E-15	-1,02E-15	0	0,00000	0,00000
2	38,8100799	6,6920946	0	0	0	0,00000	0,00000
3	58,80818276	1,080218487	0	0	0	0,00000	0,00000
4	27,85701956	5,046232585	-3,76E-16	5,38E-16	0	0,00000	0,00000
5	-17,0069384	-2,598179635	5,78E-16	8,30E-16	0	0,00000	0,00000
6	-22,6228676	-25,13073471	-1,41E-16	9,96E-17	0	0,00000	0,00000
7	57,73750537	-15,96470509	-30	-30	-30	0,00000	0,00000
8	6,531490847	-55,39368081	-30	-30	-30	0,00000	0,00000
9	48,62222097	17,80278477	0,00E+00	0,00E+00	0	0,00000	0,00000
10	-8,35062676	28,87448064	7,71E-16	-2,04E-16	0	0,00000	0,00000
11	54,83467035	15,92181382	0,00E+00	0,00E+00	0	0,00000	0,00000
12	9,051556095	-17,30031539	-12,60028781	-12,6	-13,02889051	3,28963	3,29184
13	12,20938867	-29,60569862	-29,10912512	-29,1	-28,83841509	0,93871	0,90707
14	-10,9865749	-28,24779154	-2,83E-16	-1,31E-16	0	0,00000	0,00000
15	-12,947973	-36,72352132	0	0,00E+00	0	0,00000	0,00000
16	39,72788288	-0,899596785	-30	-3,00E+01	-30	0,00000	0,00000
17	41,93361409	-2,12205868	-30	-3,00E+01	-30	0,00000	0,00000
18	52,69965271	-25,85847564	-30	-3,00E+01	-30	0,00000	0,00000
19	16,17781895	-43,83139818	-30	-3,00E+01	-30	0,00000	0,00000
20	24,81624108	-10,02565136	-19,9553879	-2,00E+01	-19,14308824	4,24331	4,47635

De la tabla se puede observar que el error promedio del FIS es 0,42358% y el de MatLab es 0,43376%.

La prueba realizada fue la prueba de comportamiento, esta se realiza comparando las superficies de decisión. La figura 27, muestra la

comparación en esta puede apreciarse que ambas describen el mismo comportamiento.

Figura 27. Comparación de la superficie de decisión FIS vs MatLab



Los resultados de las anteriores pruebas le dan credibilidad a los resultados arrojados por el componente FIS.

4.4 TERCER PROTOTIPO

En general lo que se pretende con este prototipo es corregir algunos errores encontrados durante la evaluación del prototipo anterior.

4.4.1 Análisis. Después de estudiar los errores encontrados y atendiendo a las sugerencias se tiene que:

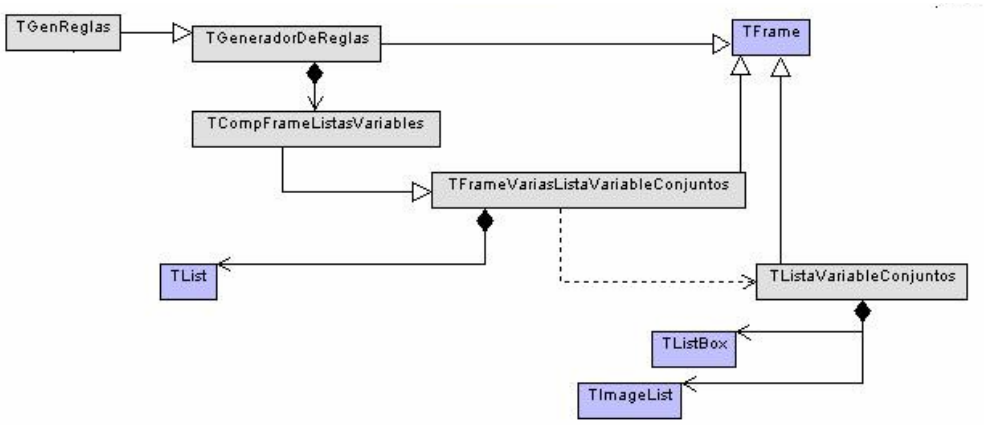
- ✓ Se debe crear un asistente que permita la edición de reglas de forma ágil, sencilla y que haga menos susceptible a errores esta tarea.
- ✓ La creación de las gráficas que representan las funciones de membresía de los conjuntos borrosos se haga con ayuda de un asistente que facilite al usuario esa tarea.

- ✓ La edición de las gráficas que representan las funciones de membresía de los conjuntos borrosos sea con ayuda de Mouse.
- ✓ Cuando se crean variables lingüísticas, se haga de tal forma que queden totalmente definidas con valores por defecto.

Debe crearse el icono que representará al elemento FIS, y asociarlo al elemento para que sea mostrado en las distintas partes que se necesite.

4.4.2 Diseño. Para cumplir con los nuevos requerimientos no fue necesario modificar la estructura de clases, solo fue necesario crear un componente **Generador de reglas** y agregarlo al editor (**TFormContenedorEditorFis**). La figura 28 muestra el diagrama de clases de este componente.

Figura 28. Generador de reglas



TGenReglas: es un componente creado a partir de la clase **TGeneradorDeReglas.**

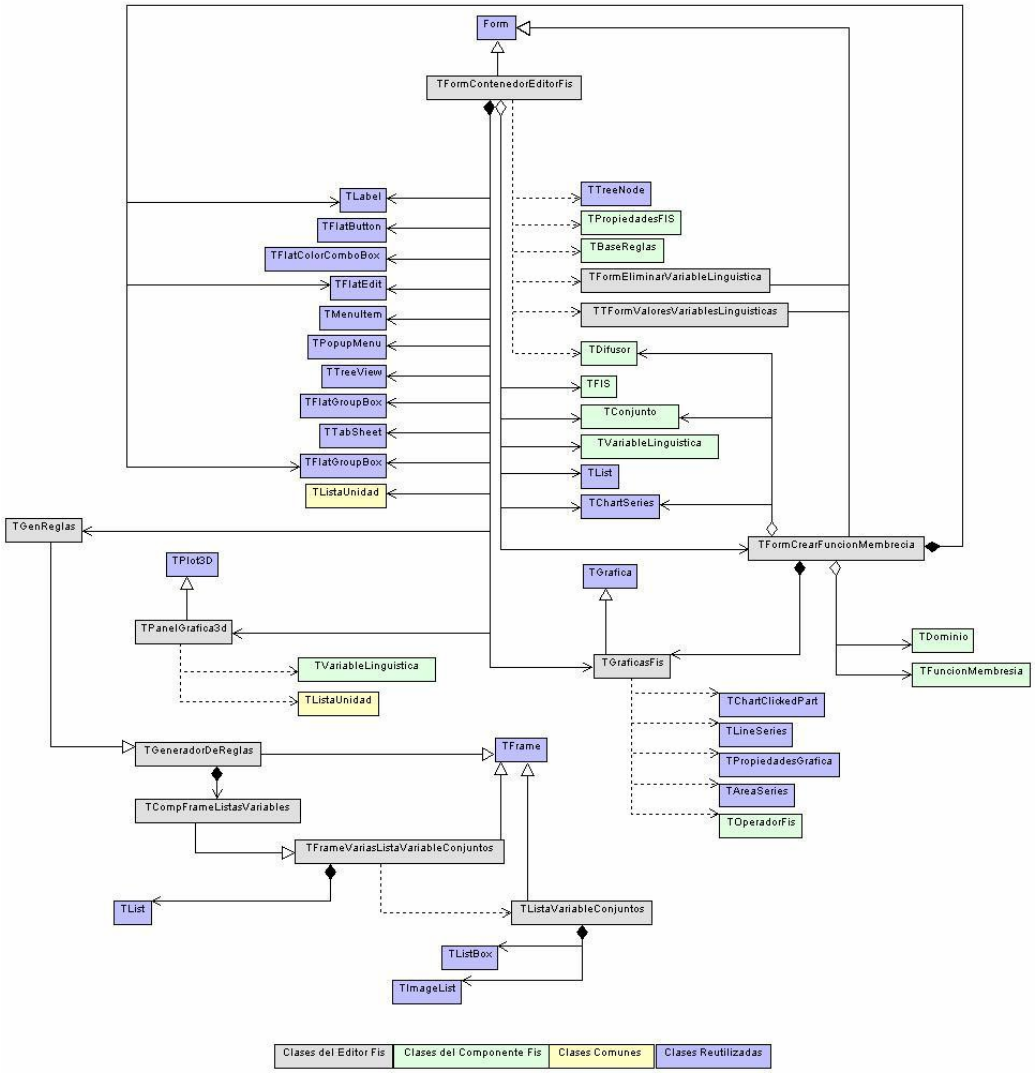
TGeneradorDeReglas: Es un asistente para la creación de reglas si – entonces, compuesta de dos elementos, uno para la salida y otro para las entradas.

TCompFrameListaVariables: es un componente creado a partir de la clase TFrameVariasListaVariableConjuntos.

TFrameVariasListaVariableConjuntos: Contiene una lista con objetos TListaVariableConjuntos, que vienen a representar los distintos conjuntos borrosos que contiene una variable lingüística.

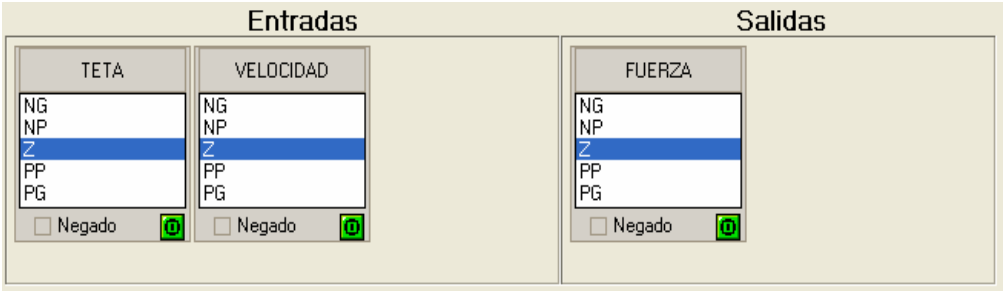
El nuevo diseño del editor (**TFormContenedorEditorFis**) se muestra en la figura 30.

Figura 29. Diagrama de clases del editor FIS.



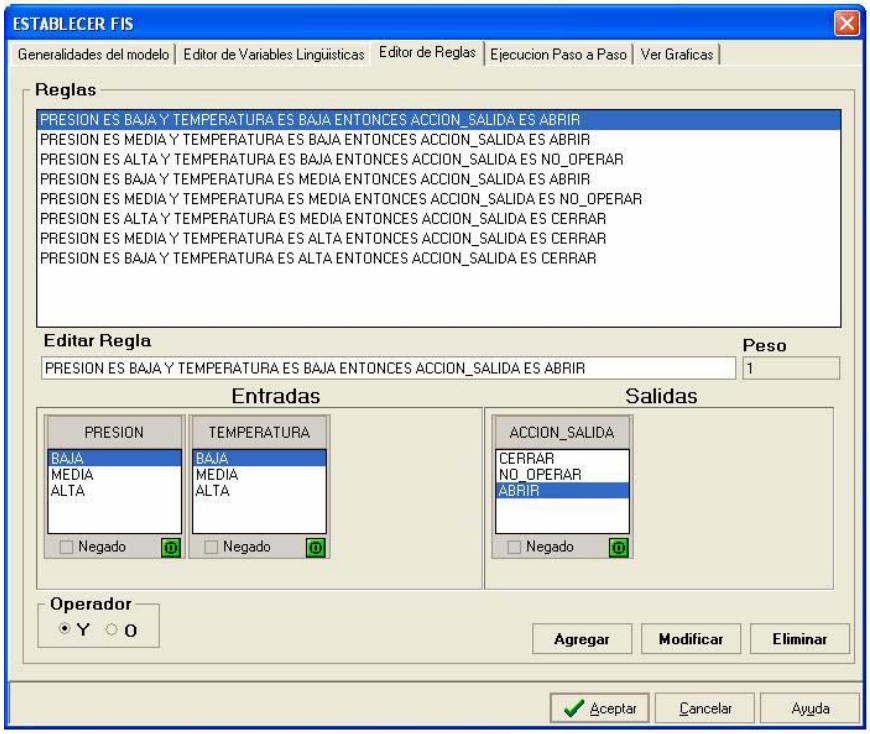
4.4.3 Implementación. Al finalizar esta fase se obtiene el nuevo componente y se suplen los requerimientos planteados en la fase de análisis. la figura 30 muestra la interfaz gráfica del asistente.

Figura 30. Asistente para generar reglas



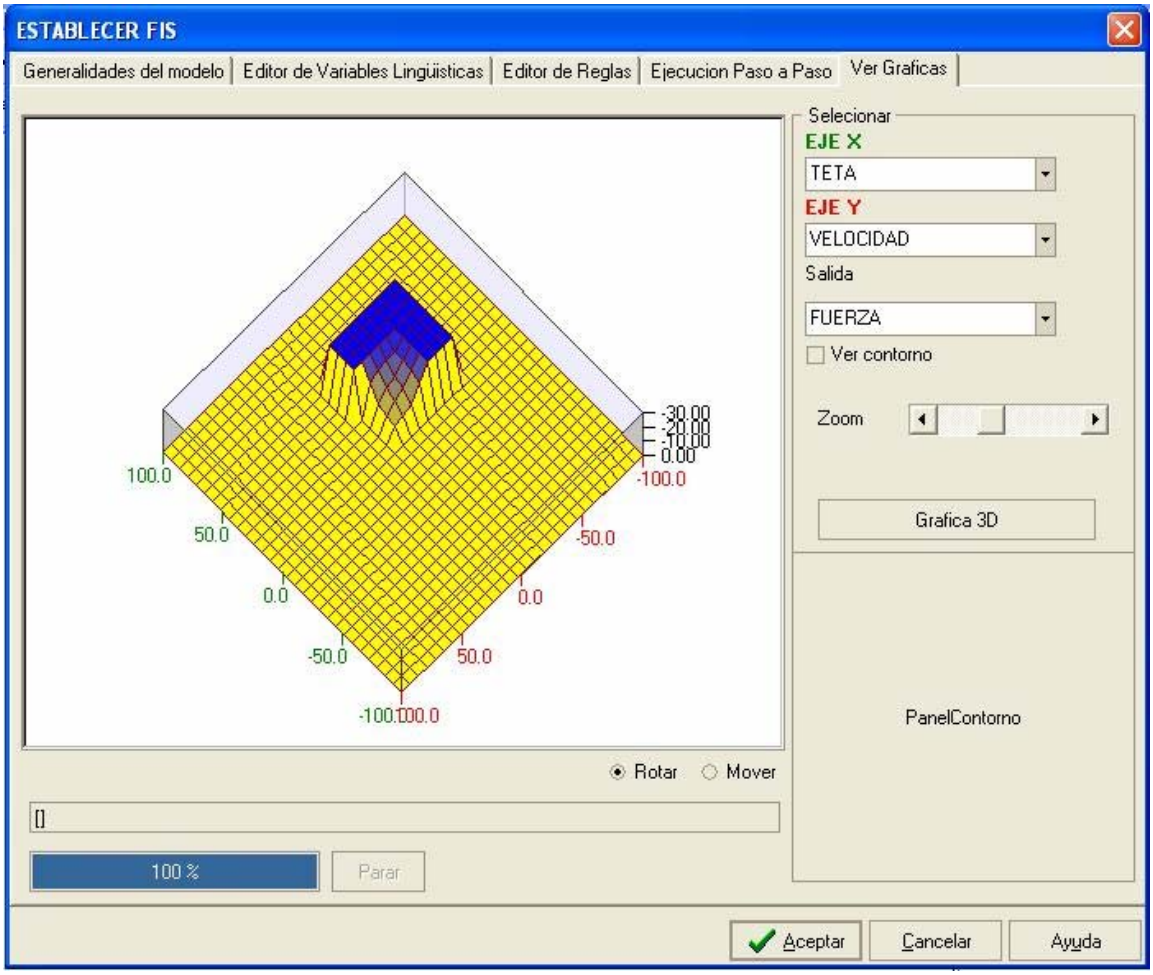
La figura 31 muestra la interfaz gráfica final del editor de reglas.

Figura 31. Editor de Reglas



La figura 32 muestra la interfaz gráfica final de la sección Ver Graficas, las diferencias con el segundo prototipo son: se agrega una barra de progreso que muestra el avance en el dibujo de la superficie de decisión; se agrega la posibilidad de mover o rotar la superficie.

Figura 32. Editor de Regla



4.4.4 Pruebas. Por ser este el último prototipo es probado exhaustivamente, dichas pruebas serán descritas en detalle en siguiente capitulo.

5. FASE PRUEBAS

5.1 INTRODUCCIÓN

Las pruebas se realizaron para verificar la calidad de los resultados entregados por el elemento FIS, así como su desempeño en con EVOLUCION. Como primer caso de prueba se utiliza el mismo ejemplo con los mismos valores del segundo prototipo, luego se utilizan otros datos para el mismo modelo y se verifica el comportamiento general con las superficies de decisión. Las pruebas descritas en este capitulo fueron aplicadas al tercer prototipo.

Dado que los resultados para los mismos valores de entrada presentados en la tabla 3 son iguales y la superficie de decisión es igual al obtenida con el segundo prototipo, se procede a probar el mismo modelo con nuevos valores. La tabla 4 muestra ese proceso.

Tabla 4. Comparación de error FIS_Evolucion vs Matlab, prototipo 3

Entradas			**Salidas**			**error**	
			FUERZA_FIS				
	TETA	**VELOCIDAD**	**FIS_Evl**	**Matlab**	**Analitico**	**Fis-Analitico**	**Matlab-Analitico**
1	14	-27	-24,66621253	-24,7	-23,70552147	4,05260	4,19513
2	12	-26	-22,5862069	-22,6	-21,6	4,56577	4,62963
3	5	-28	-22,11940299	-22,1	-21,09974425	4,83256	4,74061
4	19	-18	-17,99723757	-18	-17,51351351	2,76200	2,77778
5	25	-29	-28,52486188	-28,5	-28,10492505	1,49418	1,40571
6	13	-19	-16,1477152	-16,1	-15,93913043	1,30863	1,00927
7	10	-5	-6,710526316	-6,69	-7,674418605	12,55981	12,82727

Entradas			Salidas			error	
			FUERZA_FIS				
	TETA	VELOCIDAD	FIS_Evl	Matlab	Analitico	Fis-Analitico	Matlab-Analitico
8	12	-1	-1,698473282	-1,68	-2,171779141	21,79346	22,64407
9	1	-6	-1,493288591	-1,48	-1,917659805	22,12964	22,82260
10	22	-25	-23,53448276	-23,5	-22,57425743	4,25363	4,10088
11	14	-27	-24,66621253	-24,7	-23,70552147	4,05260	4,19513
12	12	-13	-12,86892759	-12,9	-13,22111706	2,66384	2,42882
13	1	-24	-3,771186441	-3,73	-4,621409922	18,39749	19,28870
14	26	-11	-22,16995448	-22,2	-21,19266055	4,61147	4,75325
15	11	-22	-17,36842105	-17,4	-16,93193717	2,57787	2,76438
16	15	-21	-18,42857143	-18,4	-17,85714286	3,20000	3,04000
17	11	-10	-11,12140871	-11,1	-11,72791243	5,17146	5,35400
18	3	-26	-12,46815287	-12,4	-12,98734177	3,99765	4,52242
19	25	-23	-23,63136863	-23,6	-22,6731794	4,22609	4,08774
20	19	-9	-13,51668727	-13,5	-13,79759519	2,03592	2,15686

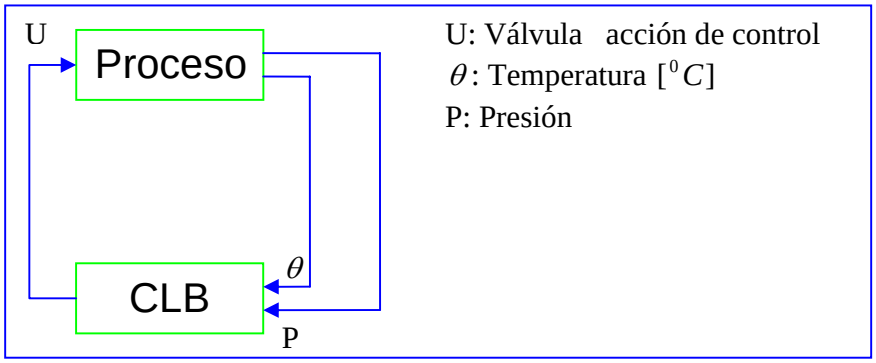
De la tabla se puede observar que el error promedio del FIS es 6,53433% y el de MatLab es 6,68721%.

5.2 EJECUCIÓN DE PRUEBAS.

A continuación se presentan una serie de ejemplos tomados de los apuntes de clases del Ingeniero Juan Carlos Reyes, profesor de la materia, Técnicas Adaptativas para el Razonamiento Aproximado, en la escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander.

EJEMPLO 2 “Modelo borroso de Mandami para dos entradas y una salida”. Se considera un proceso cuyo estado esta caracterizado por las variables lingüísticas presión y temperatura, y por una variable de acción representada por la apertura de una válvula de control, ver figura 33.

Figura 33. Lazo de control con C LB.



Variables. Las variables propias del proceso se denotan de la siguiente manera. La variable presión tiene un universo de discurso de [0 10] HPa (Hectopascales) y 3 particiones borrosas, ver figura 34. De modo similar la variable temperatura puede moverse en un rango de [50-200]^oC, con tres particiones borrosas, ver figura 35. Para la válvula de control, la apertura y cierre en porcentaje [-100 0 100] % respecto a la posición dada (valor incremental) se representa como variable lingüística según figura 36.

Figura 34. Variable Presión

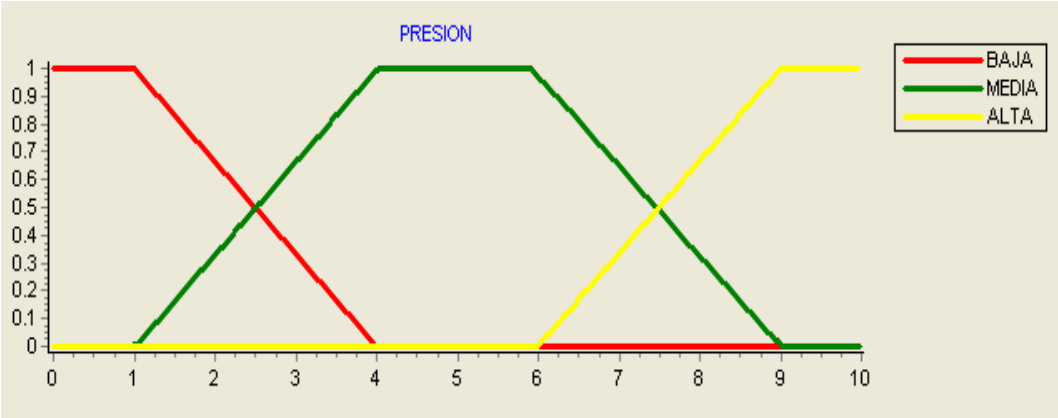


Figura 35. Variable Temperatura.

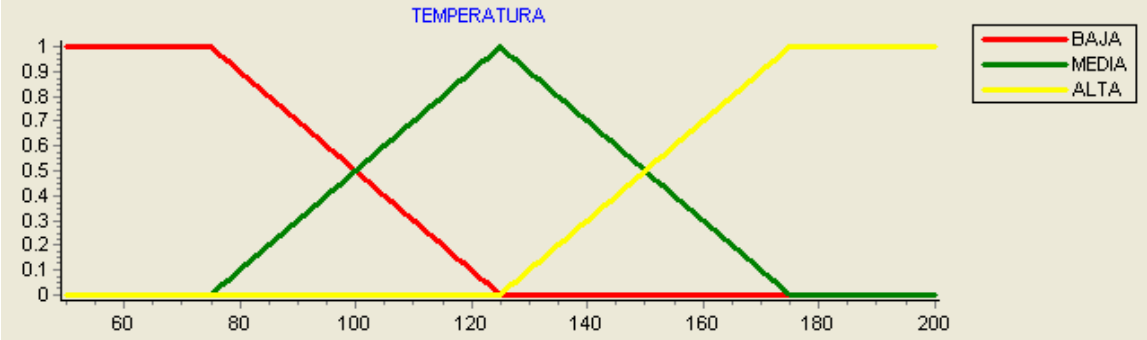
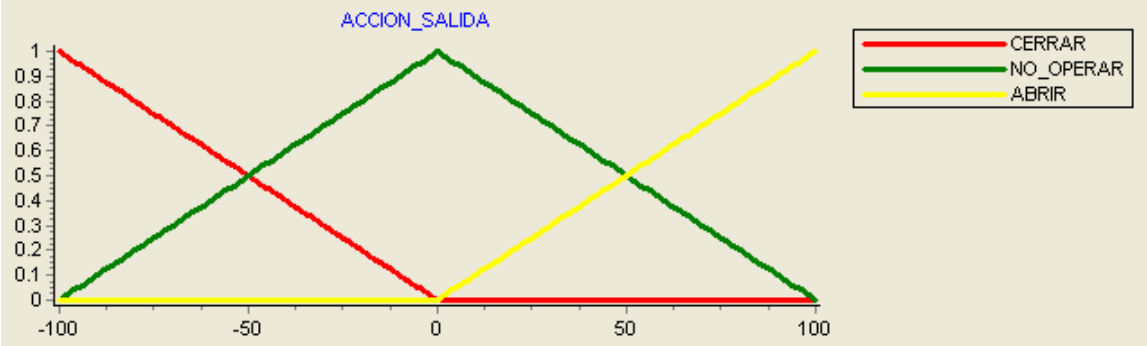


Figura 36. Variable de acción (válvula)



Base de Reglas. La base de reglas constar de hasta 9 reglas, se suponen todas las posibles combinaciones dadas por las particiones borrosas de las variables que componen el sistema.

$R_1 : Si\ \ p\ es\ baja\ Y\ \theta\ es\ baja\ \ ENTONCES\ U\ \ es\ abrir$

$R_2 : Si\ \ p\ es\ media\ Y\ \theta\ es\ baja\ \ ENTONCES\ U\ \ es\ abrir$

$R_3 : Si\ \ p\ es\ alta\ Y\ \theta\ es\ baja\ \ ENTONCES\ U\ \ No\_operar$

$R_4 : Si\ \ p\ es\ baja\ Y\ \theta\ es\ media\ \ ENTONCES\ U\ \ es\ abrir$

$R_5 : Si\ \ p\ es\ media\ Y\ \theta\ es\ media\ \ ENTONCES\ U\ \ No\_operar$

$R_6 : Si\ \ p\ es\ alta\ Y\ \theta\ es\ media\ ENTONCES\ U\ \ es\ cerrar$

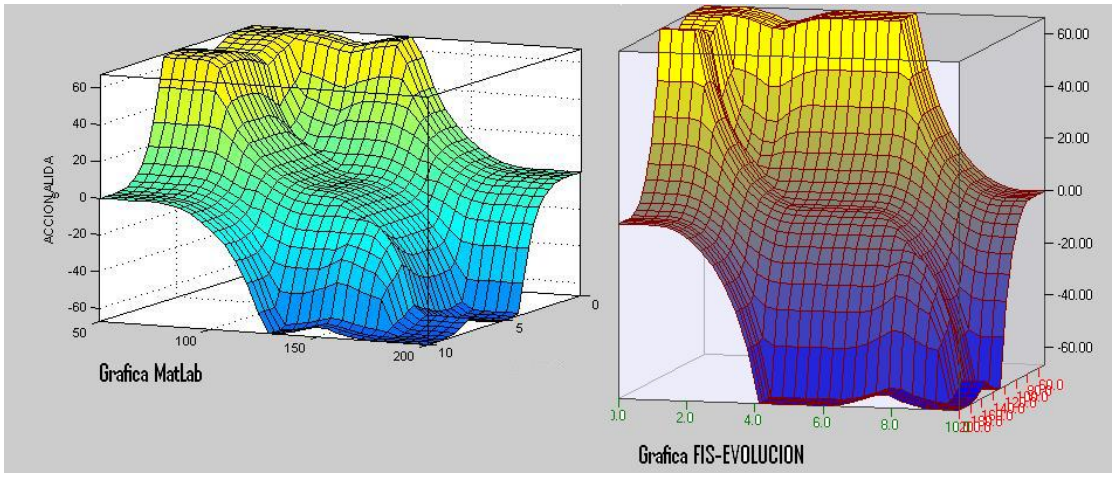
$R_7 : Si\ \ p\ es\ baja\ Y\ \theta\ es\ alta\ \ ENTONCES\ U\ \ es\ No\_operar$

$R_8 : Si\ \ p\ es\ media\ Y\ \theta\ es\ alta\ \ ENTONCES\ U\ \ es\ cerrar$

$R_9 : Si\ \ p\ es\ alta\ Y\ \theta\ es\ alta\ \ ENTONCES\ U\ \ es\ cerrar$

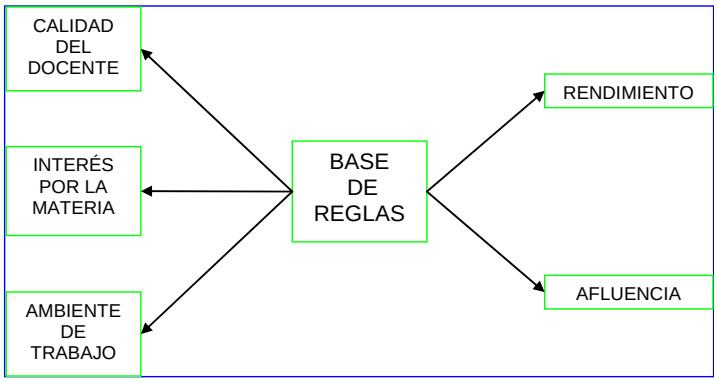
En la figura 37, se muestran las superficies de decisión de los dos modelos, en la parte izquierda se observa la obtenida con MatLab y a la derecha la superficie obtenida con el FIS de EVOLUCION.

Figura 37. Superficie de decisión. Izquierda MatLab, derecha FIS-EVOLUCION.



EJEMPLO 3. “Modelo borroso de Mandami para tres entradas y dos salidas”. Se trata de modelar la situación que se presenta comúnmente en las universidades y entidades educativas, la cuestión de cómo infieren algunos aspectos relacionados como la calidad del profesor, el interés del alumno y otros factores sobre el rendimiento del los alumnos y su asistencia a clase.

Figura 38. Esquema del ejemplo # 2



Variables. Se tendrán en cuenta tres variables de entrada y dos variables de salida. Como entradas tenemos Calidad del docente (C_prof) que tendrá tres particiones borrosas y un rango de [0 100] % ver figura 38, Interés por la materia (I_mat) con tres particiones borrosas y un rango de [0 100] % ver figura 39, y por ultimo el ambiente de trabajo (A_trab) con tres particiones borrosas y un rango de [0 100] % ver figura 40. Las salidas están conformadas por el rendimiento de los alumnos (Rend) con tres particiones borrosas y un rango de [0 100] % ver figura 41, y la afluencia de alumnos (Aflue) con tres particiones borrosas y un rango de [0 100] % ver figura 42.

Figura 38. Variable Calidad del Docente.

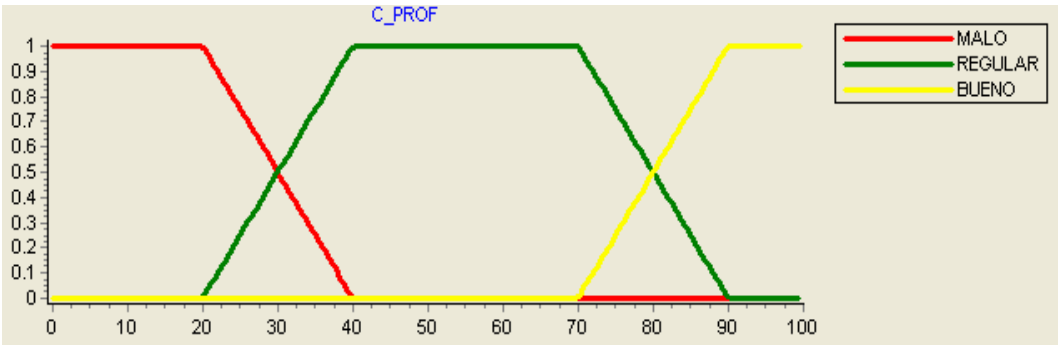


Figura 39. Variable Interés por la Materia.

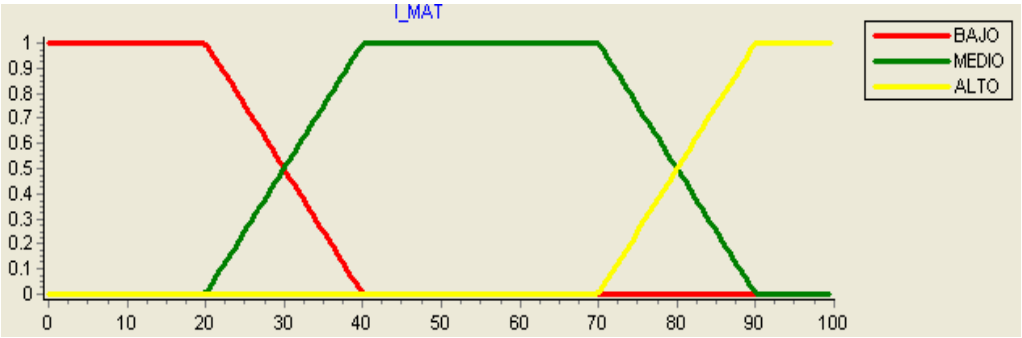
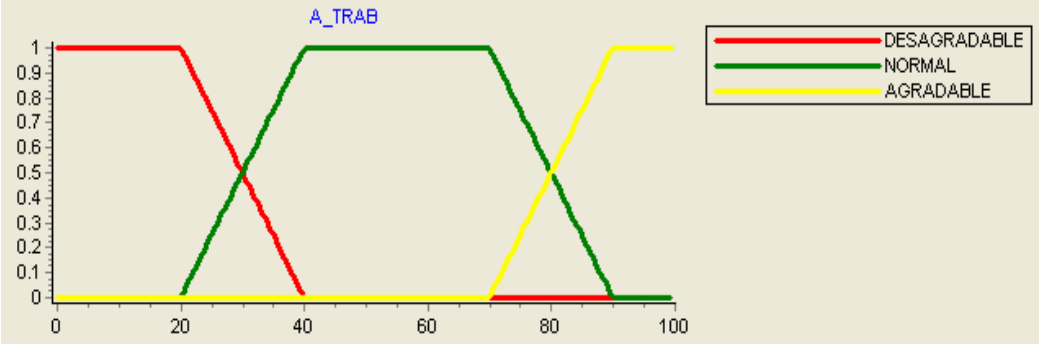
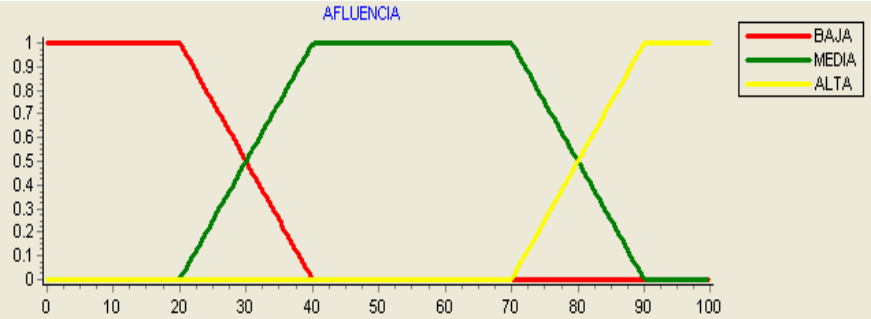


Figura 40. Variable Ambiente de Trabajo.



¡Figura 41 Variable Rendimiento.

Figura 42 Variable Afluencia.



Base de Reglas. Se tomaron todas las posibles combinaciones de las particiones borrosas a partir de las entradas, dando como resultado 27 reglas borrosas que se aprecian a continuación.

R_1 *si C_prof es malo Y I_mat es bajo Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_2 *si C_prof es regular Y I_mat es bajo Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_3 *si C_prof es bueno Y I_mat es bajo Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es media*

R_4 *si C_prof es malo Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_5 *si C_prof es regular Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es baja*

R_6 *si C_prof es buena Y I_mat es medio Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es media*

R_7 *si C_prof es malo Y I_mat es alto Y A_trab es desagradable entonces* Re*nd es malo Y Aflue es baja*

R_8 *si C_prof es regular Y I_mat es alto Y A_trab es desagradable entonces* Re*nd es aceptable Y Aflue es baja*

R_9 *si C _ prof es bueno Y I _ mat es alto Y A _ trab es desagradable entonces* Rend *es bueno Y Aflue es media*

R_{10} *si C _ prof es malo Y I _ mat es bajo Y A _ trab es normal entonces* Rend *es malo Y Aflue es baja*

R_{11} *si C _ prof es regular Y I _ mat es bajo Y A _ trab es normal entonces* Rend *es malo Y Aflue es baja*

R_{12} *si C _ prof es bueno Y I _ mat es bajo Y A _ trab es normal entonces* Rend *es aceptable Y Aflue es media*

R_{13} *si C _ prof es malo Y I _ mat es medio Y A _ trab es normal entonces* Rend *es malo Y Aflue es baja*

R_{14} *si C _ prof es regular Y I _ mat es medio Y A _ trab es normal entonces* Rend *es aceptable Y Aflue es media*

R_{15} *si C _ prof es buena Y I _ mat es medio Y A _ trab es normal entonces* Rend *es bueno Y Aflue es media*

R_{16} *si C _ prof es malo Y I _ mat es alto Y A _ trab es normal entonces* Rend *es malo Y Aflue es baja*

R_{17} *si C _ prof es regular Y I _ mat es alto Y A _ trab es normal entonces* Rend *es aceptable Y Aflue es media*

R_{18} *si C _ prof es bueno Y I _ mat es alto Y A _ trab es normal entonces* Rend *es bueno Y Aflue es alta*

R_{19} *si C _ prof es malo Y I _ mat es bajo Y A _ trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{20} *si C _ prof es regular Y I _ mat es bajo Y A _ trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{21} *si C _ prof es bueno Y I _ mat es bajo Y A _ trab es agradable entonces* Rend *es aceptable Y Aflue es media*

R_{22} *si C _ prof es malo Y I _ mat es medio Y A _ trab es agradable entonces* Rend *es malo Y Aflue es media*

R_{23} *si C _ prof es regular Y I _ mat es medio Y A _ trab es agradable entonces* Re*nd es aceptable Y Aflue es media*

R_{24} *si C _ prof es buena Y I _ mat es medio Y A _ trab es agradable entonces* Re*nd es bueno Y Aflue es alta*

R_{25} *si C _ prof es malo Y I _ mat es alto Y A _ trab es agradable entonces* Re*nd es aceptable Y Aflue es media*

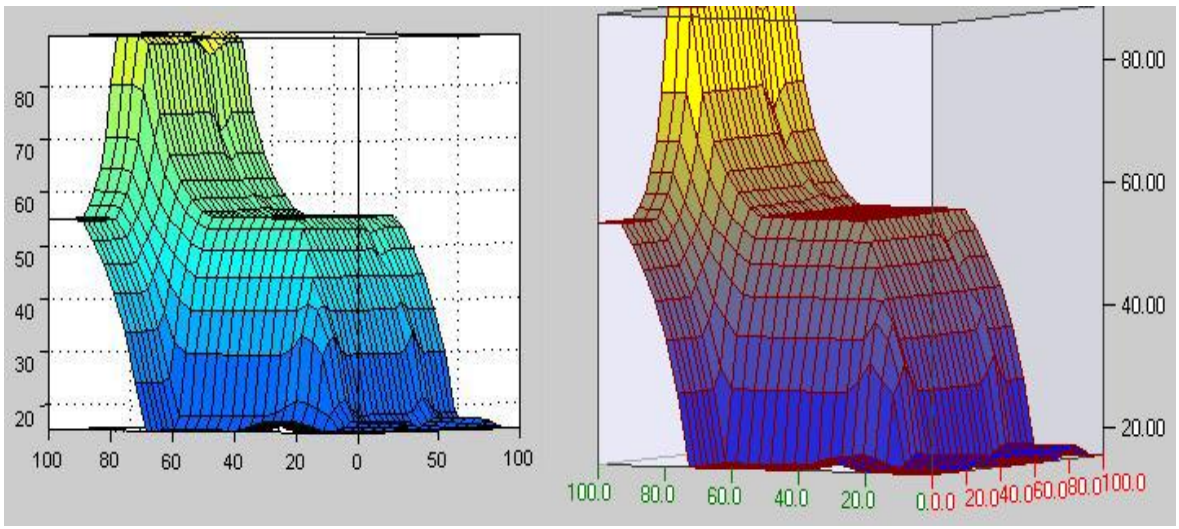
R_{26} *si C _ prof es regular Y I _ mat es alto Y A _ trab es agradable entonces* Re*nd es aceptable Y Aflue es alta*

R_{27} *si C _ prof es bueno Y I _ mat es alto Y A _ trab es agradable entonces* Re*nd es bueno Y Aflue es alta*

A continuación se comparan las superficies de decisión, obtenidas con el componente FIS y MatLab:

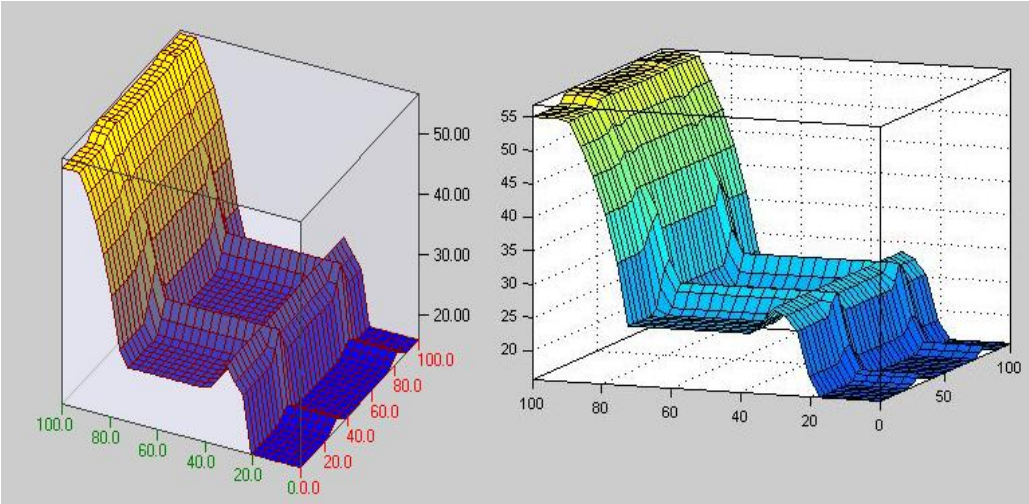
1. Calidad Profesor, Interés por la materia variables y ambiente de trabajo fijo en 50. La salida elegida es rendimiento. Ver figura 43.

Figura 43. Salida rendimiento. Izq. Matlab, der. Componente FIS-EVOLUCION.



2. Calidad Profesor, ambiente de trabajo y Calidad Profesor fijo en 22.5. La salida elegida es rendimiento. Ver figura 44.

Figura 44. Salida Rendimiento, Izquierda Componente FIS-EVOLUCION, derecha MatLab.



5.3 RESULTADOS

De las pruebas se puede concluir que el componente FIS, presenta un buen comportamiento, por dos motivos.

1. En las comparaciones con valores obtenidos analíticamente su error esta, en niveles pequeños (menores 6.7%), y en comparación con MatLab es FIS siempre gana.

2. En las figuras 27, 37, 43, 44, se puede observar que las superficies de decisiones generadas por el componente FIS, son iguales a generadas por MatLab. Si se tiene en cuenta que para generar cada superficie de decisión se utiliza una cuadricula de 30X30, en las pruebas aquí documentadas el

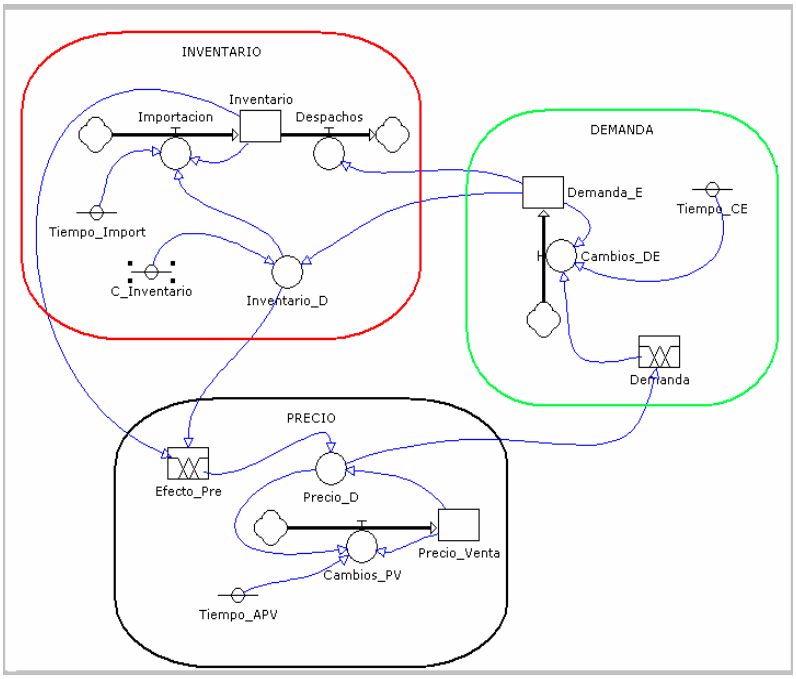
componente FIS, ha sido evaluado para 3600 valores aleatorios, ofreciendo resultados fiables y sin presentar errores en tiempo de ejecución.

A pesar de los buenos resultados arrojados por las pruebas, cabe destacar que el modulo FIS, es mucho más lento que MatLab.

5.4 EJEMPLO DE INTEGRACIÓN DE FIS CON EVOLUCIÓN

Se trata de un ejemplo sencillo donde se ilustra la manera como interactúa el elemento FIS con los demás elementos del diagrama de Forrester; es un modelo de dinámica de sistemas que representa como se maneja el precio de un producto teniendo en cuenta el nivel en existencias y la cantidad que se cree necesaria para satisfacer una demanda, y la manera como el precio afecta dicha demanda, en la figura 45 se muestra el modelo de ejemplo:

Figura 45. Modelo de dinámica de sistemas integrado con FIS



En la figura 45 se pueden ver dos elementos FIS uno llamado **efecto_Pre** el cual representa el cambio en el precio dependiendo del inventario deseado (es la cantidad que se cree es suficiente para satisfacer la demanda) y el inventario real; el segundo FIS representa el efecto del precio sobre la demanda, aunque este podría representarse con un multiplicador (que sería más eficiente computacionalemente) se optó por utilizar el FIS por el objetivo del ejemplo.

En la tabla 5 se muestran los elementos del modelo y sus definiciones sin tener en cuenta los elementos FIS:

Tabla 5. Definición de los elementos de modelo de ejemplo

Nombre del elemento	Tipo de elemento	Definición
C_Inventario	Parametro	1.5
Cambios_DE	Flujo	(Demanda_DEMANDA-Demanda_E)/Tiempo_CE
Cambios_PV	Flujo	(Precio_D-Precio_Venta)/Tiempo_APV
Demanda_E	Nivel	200
Despachos	Flujo	Demanda_E
Importacion	Flujo	(Inventario_D-Inventario)/Tiempo_Import
Inventario	Nivel	300
Inventario_D	Auxiliar	Demanda_E*C_Inventario
Precio_D	Auxiliar	Efecto_Pre_EFECTO_PRECIO*Precio_Venta
Precio_Venta	Nivel	30000
Tiempo_APV	Parametro	6
Tiempo_CE	Parametro	8
Tiempo_Import	Parametro	1

La tabla 6 muestra las características de los elementos FIS involucrados en el ejemplo.

Tabla 6. Características de los elementos FIS del modelo

Efecto_Pre			
Variable lingüística	**Conjuntos borrosos**	**Función**	**Parámetros**
variable de entrada, **Inventario_D**	bajo	Trapezoidal	[-250,0,125,250]
	estable	Triangular	[125,250,375]
	alto	Trapezoidal	[250,375,500,600]
variable de entrada, **Inventario**	bajo	Trapezoidal	[-250,0,125,250]
	estable	Triangular	[125,250,375]
	alto	Trapezoidal	[250,375,500,600]
variable de salida, **EFECTOPRECIO**	baja	Trapezoidal	[-0.5,-0.09,0.29,0.8]
	mantiene	Triangular	[0.45,0.76,1.09]
	sube	Trapezoidal	[-0.5,-0.09,0.29,0.8]
Demanda			
variable de entrada, **Precio_D**	bajo	trapezoidal	[0,30000,40000,50000]
	medio	triangular	[40000,50000,60000]
	alto	trapezoidal	[50000,60000,70000,90000]
variable de salida, DEMANDA	baja	trapezoidal	[100,150,180,225]
	estable	triangular	[180,225,270]
	alta	trapezoidal	[225,270,300,350]

La base de reglas para el elemento FIS Efecto_Pre, intentan capturar la forma como se afecta el precio de un producto, teniendo como criterio de decisión el inventario que se tiene y la cantidad que se cree necesaria para satisfacer la demanda (INVENTARIO_D), por ejemplo se intenta modelar situaciones como: si INVENTARIO de producto es BAJO y se cree que para satisfacer la demanda se necesita poco, entonces se decide no variar el precio, esto en reglas para el FIS seria (INVENTARIO ES BAJO Y INVENTARIO_D ES BAJO ENTONCES EFECTO_PRECIO ES ESTABLE). A continuación se presentan todas las reglas de este elemento:

- ✓ INVENTARIO ES BAJO Y INVENTARIO_D ES BAJO ENTONCES EFECTO_PRECIO ES ESTABLE
- ✓ INVENTARIO ES BAJO Y INVENTARIO_D ES MEDIO ENTONCES EFECTO_PRECIO ES ESTABLE

- ✓ INVENTARIO ES MEDIO Y INVENTARIO_D ES MEDIO ENTONCES EFECTO_PRECIO ES ESTABLE
- ✓ INVENTARIO ES MEDIO Y INVENTARIO_D ES BAJO ENTONCES EFECTO_PRECIO ES ESTABLE
- ✓ INVENTARIO ES BAJO Y INVENTARIO_D ES ALTO ENTONCES EFECTO_PRECIO ES SUBE
- ✓ INVENTARIO ES MEDIO Y INVENTARIO_D ES ALTO ENTONCES EFECTO_PRECIO ES SUBE
- ✓ INVENTARIO ES ALTO Y INVENTARIO_D ES BAJO ENTONCES EFECTO_PRECIO ES BAJA
- ✓ INVENTARIO ES ALTO Y INVENTARIO_D ES MEDIO ENTONCES EFECTO_PRECIO ES ESTABLE
- ✓ INVENTARIO ES ALTO Y INVENTARIO_D ES ALTO ENTONCES EFECTO_PRECIO ES ESTABLE

La base de reglas para el elemento FIS Demanda, expresan las concepciones más comunes que son: si el precio sube la demanda baja, si el precio baja la demanda sube y si el precio no cambia la demanda se mantiene, esto traducido a reglas del FIS es:

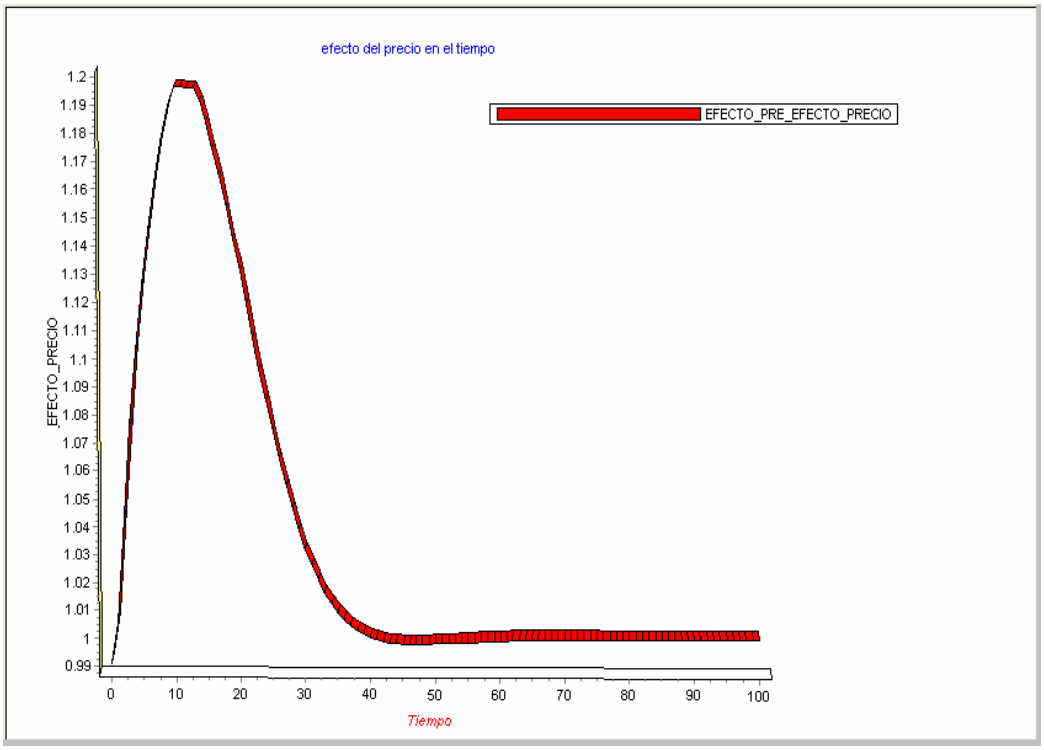
- PRECIO_D ES ALTO ENTONCES DEMANDA ES BAJA
- PRECIO_D ES MEDIO ENTONCES DEMANDA ES ESTABLE
- PRECIO_D ES BAJO ENTONCES DEMANDA ES ALTA

Los operadores son:

Máximo para la Unión
Mínimo para la Intersección
Mínimo para la implicación
Máximo para la agregación

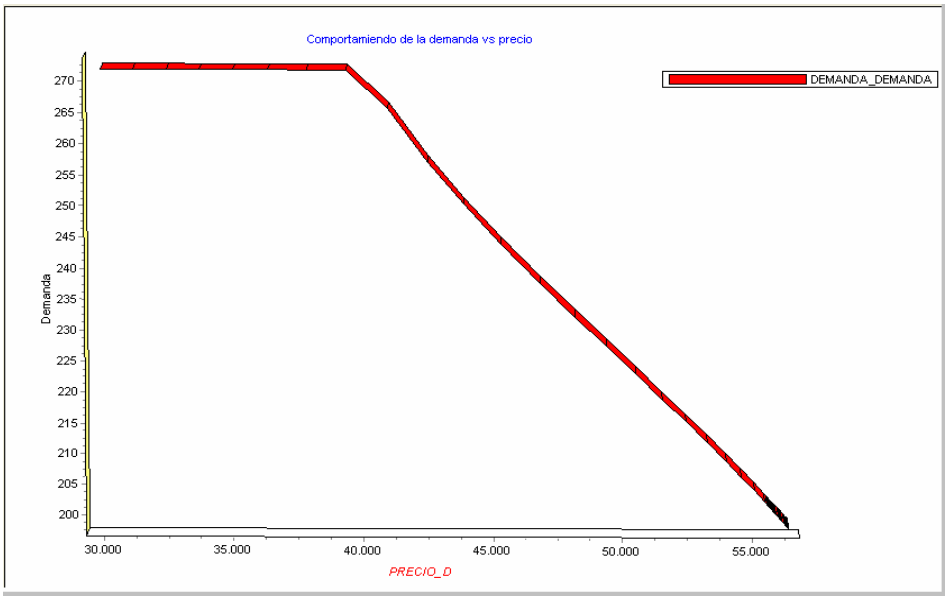
El comportamiento de la variable efecto_precio (esta variable actúa como un factor que multiplica el precio de venta para aumentarlo o disminuirlo) para una corrida de 100 iteraciones se muestra en la figura 46, se ve que el efecto sobre el precio es de aumento durante los primero 15 intervalos de tiempo, y luego disminuye hasta estabilizarse en un valor que hace que el precio se mantenga constante:

Figura 46. Presentación de resultados de simulación de una salida del FIS



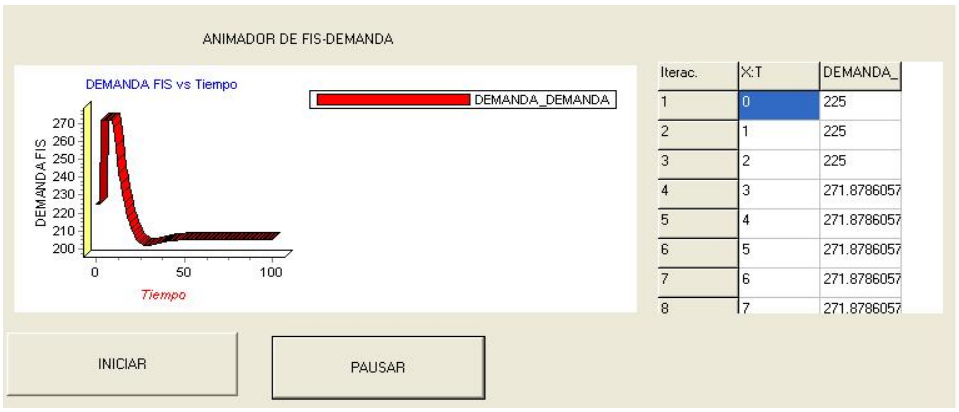
La figura 47 muestra el comportamiento de la demanda con respecto al precio, se puede notar que a medida que el precio aumenta la demanda disminuye, comportamiento normal de un producto en el mercado.

Figura 47. Demanda vs. Precio.



La figura 48, muestra un animador, en cual se observa el comportamiento de la demanda contra el tiempo en una grafica y una tabla.

Figura 48. Animador, para presentar la Demanda en el tiempo.



En este ejemplo se observó como interactúa el componente FIS, con todas las herramientas para modelado y presentación de resultados de EVOLUCION.

6. CONCLUSIONES.

- EVOLUCIÓN, posee un nuevo elemento para la construcción de modelos de Dinámica de Sistemas, el cual le permite la creación de Sistemas de Inferencia Difusa (FIS por sus siglas en Ingles), que se comporta como cualquier otro elemento del diagrama de Forrester. Dando así cumplimiento al objetivo general, planteado al iniciar este proyecto.

- Por el diseño basado en componentes, este nuevo módulo de Evolución puede ser usado en el entorno de evolución mismo o en el desarrollo de otras aplicaciones y para apoyar el trabajo de las asignaturas que requieran implementar FIS.

- El diseño del componente, y de la integración con EVOLUCION, están expresados utilizando UML, lo cual lo hace fácil de entender y lo libera de ambigüedades.

- El dotar a EVOLUCION de un elemento que permite la relación de la Dinámica de Sistemas y la Lógica Difusa, se le da la posibilidad a la comunidad que trabaja con el modelado en Dinámica de Sistemas, de abordar la complejidad de tipo cualitativo y cuantitativo de una manera sencilla.

- Ahora EVOLUCION cuenta con una arquitectura de software, que da la posibilidad, de agregar nuevos elementos que impliquen la gestión de varias salidas (ver figura 21).

- Este es otro ejemplo que muestra como la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander, se esta

en capacidad de crear herramientas software que estén a la altura de las que circulan a nivel internacional.

- Aunque el componente FIS, es fácil de usar, se requiere de unos conocimientos básicos en Lógica Difusa, para obtener mejor provecho de sus características, los cuales se resumen en la ayuda asociada a éste módulo.

- Desarrollar este proyecto requirió un estudio de temas como, Lógica Difusa, Ingeniería de Software, diseño y programación orientada a objetos. Para este proceso fue de gran ayuda la Fundamentación matemática que el Ingeniero de Sistemas de la Universidad Industrial de Santander adquiere durante su formación.

El desarrollo de este proyecto contribuyó al mantenimiento de EVOLUCION, en este sentido fueron corregidos los errores:

- En el análisis de sensibilidad por parámetros no se tenía en cuenta los parámetros que alimentaban al retardo, es decir, no se tomaba en cuenta la variación sino que se tomaba el valor que se le daba inicialmente.

- No era posible abrir un archivo de EVOLUCION que se encontraba en una ubicación de sólo lectura.

- Cuando se deseaba seleccionar todos los elementos en el diagrama de Forrester a partir del comando "*seleccionar todo*" no se tenían en cuenta los sectores.

7. RECOMENDACIONES

- En las pruebas realizadas se notó que el modulo FIS, arroja resultados mas cercanos al valor analítico, que MatLab. Sin embargo este último es más eficiente computacionalmente. Esto debido a que los algoritmos implementados para desborrosificar no eficientes. Se recomienda implementar algoritmos que mejoren esta debilidad.

- Definir una metodología que guié, el modelado con Dinámica de Sistemas, utilizando FIS, esta debe definir en que casos es factible la implementación de un FIS y en cuales no. Es recomendable el uso de los FIS, para, modelos que impliquen la razonalidad humana y en general cuando se quiere plasmar en el modelo, mecanismos que comúnmente operan con variables cualitativas.

- Implementar una interfaz que le permita al usuario definir sus propios operadores Lógicos, así como agregar nuevas funciones de membresía para la definición de los conjuntos borrosos y nuevos métodos para la agregación, desborrosificación y borrosificación.

- Realizar una valoración para determinar que tan viable desde el punto de vista conceptual, es permitir la creación de sistemas de inferencia tipo Takagi-Sugeno. Dentro del componente FIS.

- Investigar la posibilidad de dotar al FIS, de herramientas que permitan su entrenamiento para así lograr su funcionamiento óptimo. Para lograr este se

pueden utilizar **Redes Neuronales Artificiales, Algoritmos Genéticos,** entre otros.

- Hacer un uso intenso de esta herramienta, tanto para simulaciones dentro de EVOLUCIÓN, como en el desarrollo de otro software. Esto con el fin de recolectar errores, para corregirlos y así lograr una herramienta estable.

BIBLIOGRAFÍA

- AJA FERNÁNDEZ, Santiago, Tesis doctoral Un nuevo marco matricial para la implementación de inferencia borrosa aplicados al procesado de información no numérica, Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, Escuela Técnica Superior de Ingenieros de Telecomunicaciones, Universidad de Valladolid. 2003.

- ANDRADE, Hugo Hernando; DYNER, Isaac; ESPINOSA, Ángela, LOPEZ GARAY, Hernán y SOTAQUIRA, Ricardo. Pensamiento Sistémico: Diversidad en búsqueda de unidad. 2001.

- ARACIL, Javier. Introducción a la Dinámica de Sistemas. Alianza Editorial. Madrid, 1986.

- AUGUSTO RIVERA, César; SARMIENTO VILLAMIZAR, Freddy; Andrade Sosa, Hugo Hernando. INTEGRACIÓN ENTRE LA DINÁMICA DE SISTEMAS Y OTRAS MATEMÁTICAS PARA LA REPRESENTACIÓN DEL CONOCIMIENTO

- AUGUSTO RIVERA, César; SARMIENTO VILLAMIZAR, APLICACIONES DE UN “MODELO CONCEPTUAL GENERALIZADO DE UN SISTEMA PARA SOPORTAR LA TOMA DE DECISIONES”, Enfoque Integrador Entre la Dinámica de Sistemas y la Inteligencia Artificial. 2005.

- BERTALANFFY, Ludwing Von. Teoría General de Sistemas. Fondo de Cultura Económica, 1976.

- CUELLAR YEMERIS, Mario; LINCE MERCADO, Emiliano; EVOLUCIÓN 3.5: HERRAMIENTA SOFTWARE PARA EL MODELAMIENTO Y SIMULACIÓN EN DINAMICA DE SISTEMAS, Proyecto de grado, Bucaramanga, 2003.

- DUARTE VELASCO, Óscar Germán. "UNFUZZY 1.2: Herramienta de Software para el Análisis, Diseño, Simulación e Implementación de Sistemas de Lógica Difusa". Universidad Nacional de Colombia, sede Bogotá (1998). Información disponible en: http://www.ing.unal.edu.co/~electrica/Proyectos/Proyecto5.html

- HILERA José R; MARTÍNEZ Víctor. Redes Neuronales Artificiales Fundamentos, módulos y aplicaciones, ALFAOMEGA RA-MA. 2000.

- JOYANES, Luis. Programación orientada a objetos. McGraw-Hill. Madrid. 1998.

- KLIR/BO YUAN; J, GEORGE. Fuzzy Sets and fuzzy Logic Theory and Applications, Prentice Hall. 1995.

- LUQUE, Guillermo; PRADILLA, Vladimir. Modelo Conceptual Generalizado de un Sistema para Soportar la Toma de Decisiones: Enfoque Integrador Entre la Dinámica de Sistemas y la Inteligencia Artificial, Proyecto de grado, Bucaramanga, 2003.

- MARTÍN DEL BRÍO, Bonifacio; SANZ MOLINA, Alfredo. Redes Neuronales y Sistemas Difusos, 2ª Edición, ALFAOMEGA RA-MA. 2001.

- MEYER, Bertrand. Construcción de software orientado a objetos. 2 ed. Madrid. Prentice Hall. 1998.

- PRESSMAN, Roger. Ingeniería Del Software Un Enfoque Practico. McGraw Hill. Tercera Edición. 1993.

- RANBAUG, James, Jacobson Ivan, Booch Grady. El Lenguaje Unificado de Modelado - Manual de Referencia. Pearson Educación, S.A, Madrid, 2000.

ANEXOS

Anexo A. DIAGRAMAS DE SECUENCIA

Diagrama de secuencia preparar FIS.

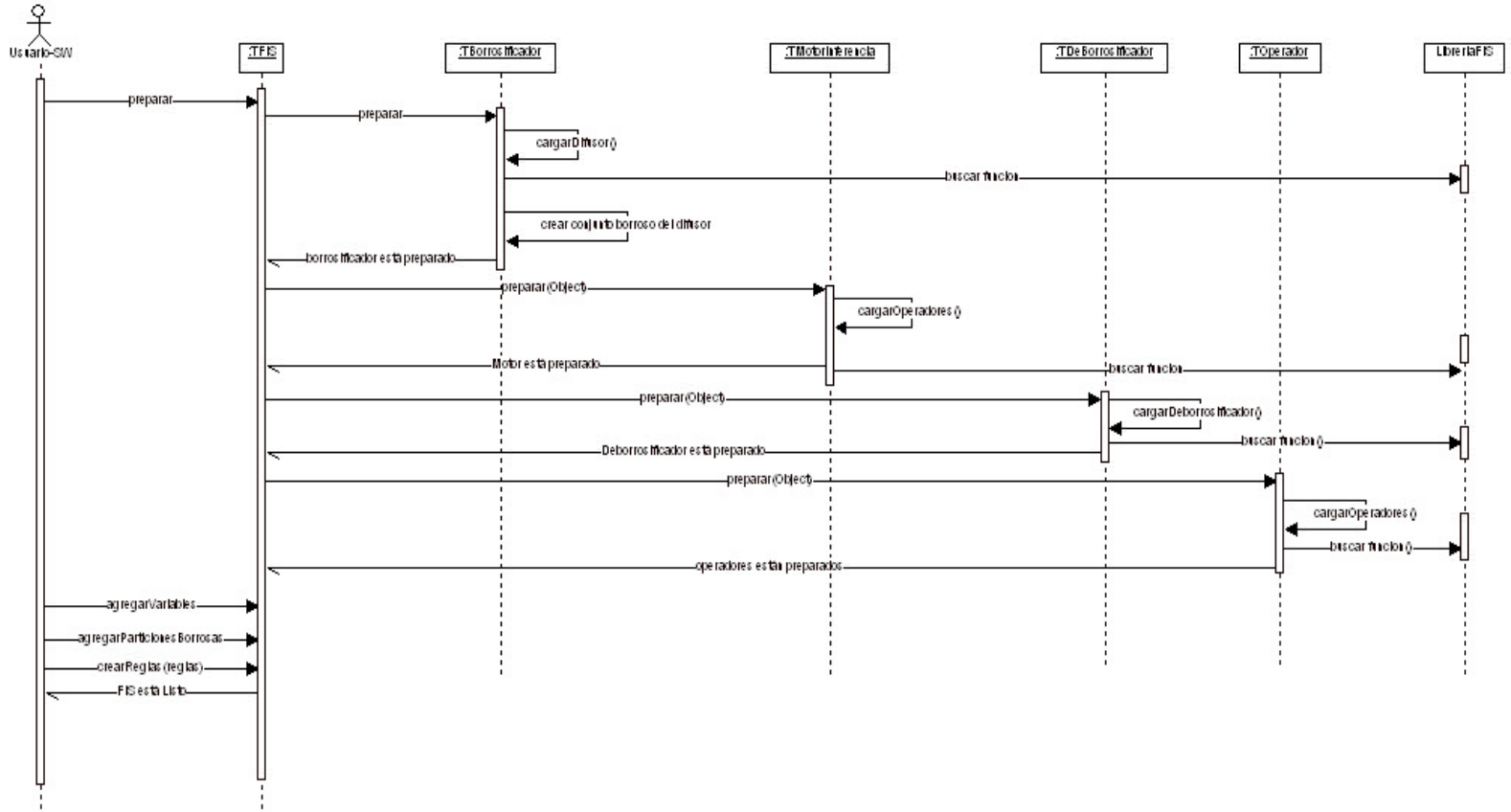


Diagrama de secuencia Iniciar FIS.

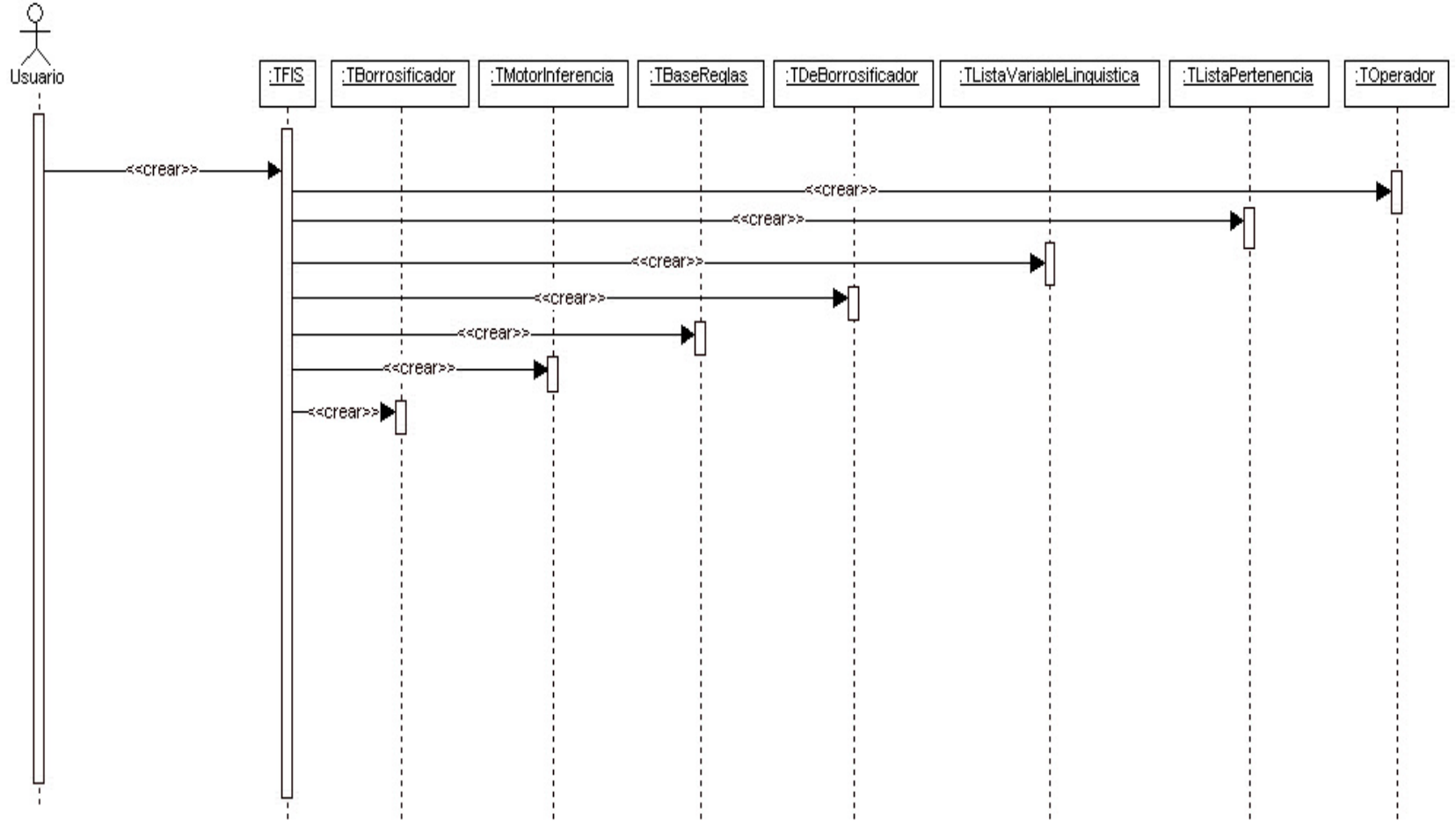


Diagrama de secuencia Evaluar FIS.