Universidad de las Ciencias Informáticas

Facultad 3

Diseño de un algoritmo basado en GenRul5 para el aprendizaje automatizado de reglas difusas

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Eden Sobrino Quiñones

Tutor: Ing. Carlos Rafael Rodríguez Rodríguez

La Habana, julio de 2014

El sacrificio forja al hombre así como el hábito educa al monje.

E.S.

Agradecimientos

En estos momentos si tuviese que compartir el valor de mi título, creo que no me quedaría ni un centavo. Muchas han sido las personas que me han ayudado a cumplir ese gran sueño de convertirme en ingeniero.

Primeramente debo agradecer a mi papá, muchas gracias por estar siempre al pendiente, por ser mi maestro de todos los días. ¿Te acuerdas cuando me enseñaste a tender una cama en la beca? Disculpa cada dolor de cabeza que te di, gracias mostrarme que nunca son suficientes los besos y abrazos entre un padre y su hijo. Quisiera cuando tenga mi familia poder darle todo eso que he recibido de ti.

Mami ahora mismo me pregunto ¿qué habría sido de mí? sin ti. Gracias por estar siempre ahí. Gracias por ser paciente, por darme seguridad y esperanzas cuando más las he necesitado. Espero algún día poder compensar todos los sacrificios que tanto tú como mi papá han hecho por mí.

Muchas gracias a mis abuelos, tíos y primos que no dejaron de rezar por mí.

Debo agradecer a mi tutor por haberme confiado parte de su tesis de maestría. Gracias por todas esas horas de trabajo conjunto, por cada señalamiento en busca de un buen trabajo de diploma.

Muchas gracias a todos los profesores y trabajadores con los que he compartido y me han brindado su apoyo tanto en la universidad como en enseñanzas precedentes a lo largo de estos 17 años de estudio.

Finalmente quiero agradecer a los compañeros de aula y amigos de la universidad con los que he compartido esta magnífica etapa de mi vida y se han convertido en mi otra familia.

A todos Muchas Gracias.

Dedicatoria

Le dedico el fruto de mis años de estudio a la única persona que hoy no podré abrazar y decirle "lo logré". Va dedicado a mi abuelo paterno que desde sus limitaciones siempre tuve su apoyo y hace algunos años no se encuentra presente entre nosotros.

Declaración jurada de autoría

Declaramos ser autores de la presente tesis y le conferimos con carácter permanente el derecho de uso para el desarrollo institucional a la Universidad de las Ciencias Informáticas en consecuencia con los principios del desarrollo y la moral socialista.

Para que así conste firmo la presente a los ____ días del mes de julio del año 2014.

Eden Sobrino Quiñones

_____

Firma del Autor

Carlos Rafael Rodríguez

_____

Firma del Tutor

Resumen

La suite de Gestión de Proyectos (GESPRO) es un sistema informático para la dirección integrada de proyectos que toma como base los modelos planteados por el Project Management Body of Knowledge (PMBOK). Sin embargo, existen deficiencias al tomar decisiones con ayuda de dicho entorno pues para ello emplea un sistema de inferencia difuso donde la base de reglas tiene la particularidad de haber sido especificada de forma manual siguiendo el criterio de expertos. Esta característica trae implícita la presencia de incertidumbre en las valoraciones realizadas y ambigüedad en los conceptos empleados, lo que podría desencadenar la obtención de resultados que no cuenten con la precisión deseada y de esa forma se afectaría la efectividad del proceso de toma de decisiones. El objetivo de la investigación es diseñar un algoritmo para el aprendizaje automatizado de reglas difusas, que permita mejorar la capacidad de ayuda a la toma de decisiones de la herramienta GESPRO. El resultado obtenido fue la implementación para la explotación en GESPRO de un conjunto de funcionalidades del diseño de un algoritmo basado en GenRul5 para el aprendizaje automatizado de reglas de inferencia difusa.

Palabras claves: aprendizaje automatizado, generación de reglas, GenRul5, lógica difusa.

Abstract

The Project Management suite (GESPRO) is an integrated informatic system for project management which is based on the models proposed by PMBOK. There are deficiencies to make decisions using that environment to do it employs a fuzzy inference system where the rule base has the distinction of being manually specified according to the criteria of various experts. This feature brings implied the presence of uncertainty in valuations and ambiguity in the concepts used, which could trigger obtaining results that do not have the desired precision and thus the effectiveness of the decision-making process would be affected. The objective of the research is to implement an algorithm for automated learning of fuzzy rules, which will improve the ability to aid the decision-making tool GESPRO. The result was the implementation for exploitation in GESPRO of a set of features of a version of GenRul5 algorithm for automated learning of fuzzy inference rules.

Key words: machine learning, generation of rules, GenRul5, fuzzy sets.

Índice

Introducción

Las características actuales de la economía global requieren de organizaciones cada vez más eficaces y eficientes en su desempeño, que tengan la capacidad de regenerarse continuamente, dando un lugar privilegiado al conocimiento organizacional (Castro Díaz-Balart, y otros, 1999). Así como los recientes Lineamientos de la Política Económica y Social del Partido y la Revolución aprobados en el sexto congreso del Partido Comunista de Cuba (PCC), exigen mecanismos eficientes de control que permitan dar seguimiento al estado de los proyectos en las organizaciones y contribuir a la toma de decisiones (PCC, 2011). Todo ello podría entenderse como la capacidad de mantener una alta competitividad ante los avances de la ciencia y la tecnología.

La Gestión de Proyectos no exenta de competitividad, y necesitada de mecanismos idóneos para obtener un resultado satisfactorio definen el proceso de "*administrar los recursos humanos y materiales, a lo largo del ciclo de vida del proyecto, para conseguir los objetivos de alcance, costos, plazo, calidad y satisfacción de los participantes*" (PMI, 2012) como Dirección Integrada de Proyectos (DIP). Con el propósito de exponer pautas para lograr una mejor dirección de proyectos, varias instituciones y autores han formulado guías, estándares y metodologías para efectuar la DIP, algunas de las más reconocidas son: ISO 21500 (ISO, 2012), PMBOK (PMI, 2012), CMMI (SEI, 2010). A pesar de que la literatura reporta numerosos estudios y aportes teóricos sobre lo que es la dirección en términos generales, todos concuerdan en que el control y la toma de decisiones juegan un rol esencial.

En el campo de la industria del software cubana, la Universidad de las Ciencias informáticas (UCI) aparece como una institución de características muy peculiares. En esta institución se integran las actividades de formación académica de pre y postgrado, la investigación y el desarrollo de software. Con el aumento de sus compromisos productivos y la magnitud de los productos a desarrollar surge la necesidad de una herramienta informática para la gestión de todos los proyectos de la amplia red de centros que actualmente posee la universidad.

Como respuesta a la necesidad existente surge el paquete de software libre para la ayuda a la toma de decisiones y la dirección integrada de proyectos GESPRO (Piñero Pérez, y otros, 2010). Esta suite, basada en Redmine y otras herramientas libres, se rige según (Piñero Pérez, 2013) por el estándar PMBOK (PMI, 2013) y las buenas prácticas del programa de formación del Máster en Gestión de Proyectos de la Universidad de las Ciencias Informáticas (Piñero, y otros, 2008). Para efectuar la toma de decisiones GESPRO implementa varias técnicas de

softcomputing como son los sistemas de inferencia difusos (SIB), los que requieren de una base de reglas para su funcionamiento.

Existen tres tipos de SIB fundamentales (Mamdani, Sugeno y Tsukamoto) diferenciados en el consecuente de las reglas que emplea. Actualmente la base de reglas que emplea GESPRO en su modelo para el control de la ejecución de proyectos está compuesta por 147 reglas, de ellas 120 pertenecientes a un SIB de tipo Mamdani y otras 27 reglas difusas para un SIB Sugeno Grado Cero (Lugo García, y otros, 2012).

Estas reglas tienen la particularidad de haber sido especificadas de forma manual siguiendo el criterio de varios expertos. Esta característica trae implícita la presencia de incertidumbre en las valoraciones realizadas por los expertos y ambigüedad en lo conceptos empleados, lo que podría desencadenar la obtención de resultados que no cuenten con la precisión deseada y de esa forma se afectaría la efectividad del proceso de toma de decisiones.

En busca de conocer las tendencias actuales de las herramientas de gestión de proyectos, se muestra en Anexo 9 un estudio comparativo realizado por el autor en octubre de 2013 sobre la suite GESPRO (Piñero Pérez, y otros, 2010) y otras 99 herramientas para la gestión de proyectos reportadas en (Stang, 2010). En dicho estudio se observa: un 11% de soluciones libres, el 90% brindan reportes y facilidades para el análisis y solo el 1% hace uso de técnicas de softcomputing para el tratamiento de la incertidumbre en la toma de decisiones, pero no para aprender las reglas que emplea.

Como se expuso con anterioridad, la suite GESPRO contiene sistemas difusos donde se puede ver la base de reglas como el centro de estos sistemas. Si la base de reglas es construida solo a partir del conocimiento adquirido previamente por un experto o grupo de expertos podría usualmente no funcionar correctamente cuando es aplicado. Esto puede ocurrir porque los expertos podrían equivocarse acerca de la localización de determinados puntos característicos en las funciones de pertenencia, respecto al número de reglas o respecto a si son distinguibles o no determinadas áreas del espacio de búsqueda (Piñero Pérez, 2005).

Teniendo en cuenta los limitaciones descritas anteriormente para la obtención de reglas difusas se establece como **problema a resolver** que: el uso de reglas difusas construidas manualmente y a partir del conocimiento previo de expertos, está limitando el completo funcionamiento de los SIB empleados en la suite GESPRO y por tanto la capacidad de ayuda a la toma de decisiones de esta herramienta.

Objeto de estudio: El proceso de aprendizaje de reglas difusas.

Campo de acción: El proceso de desarrollo de soluciones informáticas para la toma de decisiones en la DIP basada en técnicas de softcomputing.

Objetivo general: Diseñar un algoritmo para el aprendizaje automatizado de reglas difusas basado en GenRul5, de manera que permita su explotación mediante la suite GESPRO y así mejorar la capacidad de ayuda a la toma de decisiones de esta herramienta.

Objetivos específicos:

- Elaborar el marco teórico de la investigación haciendo uso de métodos empíricos, teóricos y particulares para determinar las principales tendencias en el campo del aprendizaje automatizado y su aplicación a la gestión de proyectos.
- Implementar en el lenguaje procedural de R para postgreSQL las funcionalidades de una versión del algoritmo GenRul5 de manera que permita aprender reglas a partir de una base de casos.
- Validar los resultados obtenidos a través de su aplicación a una base de casos de proyectos terminados en la Universidad de Ciencias Informáticas.

Idea a defender:

Si se diseña un algoritmo para el aprendizaje automatizado de reglas difusas, de manera que permita su explotación mediante la suite GESPRO se contribuirá a mejorar la capacidad de ayuda a la toma de decisiones de esta herramienta.

Tareas a desarrollar:

i. Investigación sobre el estado actual en Cuba y el mundo del uso del aprendizaje automatizado en la gestión de proyectos.

ii. Identificación de la necesidad y relevancia del aprendizaje automatizado para la toma de decisiones en GESPRO.

iii. Revisión de varios epígrafes de la tesis *"Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing"* del Dr. Pedro Piñero para determinar:

- ¿Qué es el aprendizaje automatizado?
- ¿Qué son los Sistemas Basados en el Conocimiento de la Inteligencia Artificial (IA)?
- ¿Qué son los SIB?
- Estrategias de generación automatizada de reglas difusas.

iv. Confección un mapa conceptual donde relacione lo aprendido antes.

v. Revisión de varios epígrafes de la tesis de doctorado "*Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing"* para determinar semejanzas y diferencias entre los algoritmos MLRul y GenRul5.

vi. Revisión bibliográfica sobre lo que es un reducto y cómo se construye.

vii. Análisis detallado de cada uno de los pasos del algoritmo GenRul5.

viii. Definición y caracterización de la notación y/o lenguaje de modelado a utilizar. Seleccionar la metodología de desarrollo de software a emplear.

ix. Definición y caracterización de las herramientas y tecnologías a utilizar para el desarrollo de la propuesta de solución.

x. Elaboración del modelo de datos.

xi. Implementación de las funcionalidades de la versión del algoritmo especificada.

xii. Verificación y validación de la solución propuesta.

xiii. Solución de no conformidades.

Métodos teóricos empleados:

Análisis y síntesis: se utiliza principalmente en la precisión de los fundamentos teóricos relacionados con la generación de reglas difusas. En el análisis, la valoración y el conocimiento de las particularidades del proceso de generación de reglas difusas y las herramientas empleadas para ello.

Inducción y deducción: este método permite analizar el proceso de generación de reglas difusas partiendo de las características propias de las reglas existentes en la base de reglas que emplea GESPRO.

Resultados esperados:

- Historias de usuario
- Lista de reserva de productos
- Tarjetas Clase-Responsabilidad-Colaborador (CRC)
- Tareas de ingeniería
- Resultados de las pruebas aplicadas al código

La presente investigación comprende la introducción, tres capítulos, las conclusiones generales, las referencias bibliográficas y los anexos. Los contenidos abordados en los tres capítulos se describen brevemente a continuación:

Capítulo 1. FUNDAMENTOS TEÓRICOS: se expone el marco conceptual de la propuesta de solución, introduciendo conceptos e ideas que se manejarán a lo largo de la investigación. Comprende el estudio del estado del arte de las herramientas y algoritmos que permiten el descubrimiento de reglas difusas. Además, se caracterizan las herramientas, la metodología y los lenguajes de programación que serán utilizados en el desarrollo de la solución.

Capítulo 2. PROPUESTRA DE SOLUCIÓN: comprende la implementación del algoritmo. Enmarca la descripción de la solución propuesta. En un primer momento se enuncian las características del algoritmo para luego modelar el mismo haciendo uso de la metodología de desarrollo de software y se presentan los principales artefactos generados en las fases que esta propone.

Capítulo 3. VERIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA: finalmente se evalúa la solución a través de los resultados obtenidos una vez realizadas las pruebas a las funcionalidades implementadas correspondientes al algoritmo y la solución de no conformidades en caso de que surjan en la etapa de validación.

Capítulo 1. FUNDAMENTOS TEÓRICOS

En el presente capítulo se exponen conceptos y términos que resultan relevantes y serán utilizados en el desarrollo de la investigación, tales como: el aprendizaje automatizado, los sistemas basados en el conocimiento de la inteligencia artificial y las estrategias de generación de reglas de inferencia difusa. También se justifica la elección de la metodología por la cual se regirá el equipo de trabajo así como las tecnologías necesarias en el modelado y el desarrollo.

1.1 Fundamentos teóricos de la investigación

La definición de aprendizaje según el diccionario de la Real Academia de la Lengua Española contempla acepciones tales como *"para ganar conocimiento o entendimiento"*, *"habilidad en el estudio de instrucciones", "ganar en experiencia"* y *"modificación de la tendencia situacional por la experiencia*" (RAE, 2014). Psicólogos, estudiosos de la zoología y tecnólogos históricamente han estudiado el aprendizaje en animales y humanos.

En busca de ganar experiencia y optimizar procesos manuales por medio de la informatización de los mismos se ha hecho necesario dotar a los equipos de cómputo de aprendizaje automatizado.

1.1.1 ¿Qué es el aprendizaje automatizado?

La mayoría de las actuales técnicas del aprendizaje automatizado a partir de modelos computacionales se derivan del aprendizaje biológico (Piñero Pérez, 2005).

Según Tom Mitchell: el aprendizaje automatizado implica la búsqueda en un gran espacio de posibles hipótesis para determinar una que sea la que mejor satisfaga los datos observados y algún conocimiento previo del aprendiz (Mitchell, 1997).

Por su parte Oliver G. Selfridge presenta el aprendizaje automatizado como la capacidad de los sistemas de integrar y adquirir conocimiento a partir de la experiencia y la observación analítica. Esta capacidad permite que los sistemas estén continuamente en un proceso de mejora incrementando su eficiencia y efectividad (Selfridge, 2004).

Existen varios modelos matemáticos para desarrollar el aprendizaje automatizado, en ellos aparecen deficiencias tales como: ruido, ambigüedad, incompletitud e imprecisión. Como solución a estos efectos negativos, se desarrollaron las técnicas de softcomputing (Zadeh,

1994), dentro de las que se pueden mencionar: las redes neuronales artificiales, la computación evolutiva y los sistemas difusos (Piñero Pérez, 2005).

1.1.2 Tipos de aprendizajes automatizados

Diversos autores coinciden en clasificar las estrategias de aprendizaje en dos tipos: los supervisados y los no supervisados. En general hay una correspondencia entre el tipo de entrenamiento y el tipo de problema a resolver (Basogain Olabe , 2009) (Selfridge, 2004).

Aprendizaje Supervisado: estos algoritmos requieren el emparejamiento de cada vector de entrada con su correspondiente vector de salida. El aprendizaje consiste en presentar un vector de entrada al modelo, calcular la salida del mismo, compararla con la salida deseada, y el error o diferencia resultante se utiliza para realimentar la red y cambiar los pesos de acuerdo con un algoritmo que tiende a minimizar el error (Basogain Olabe , 2009).

Aprendizaje No Supervisado: los sistemas no supervisados son modelos de aprendizaje más lógicos en los sistemas biológicos. Desarrollados por Kohonen (Kohonen, 1988) y otros investigadores, estos sistemas de aprendizaje no supervisado no requieren de un vector de salida deseada. El conjunto de vectores de entrenamiento consiste únicamente en vectores de entrada. El algoritmo de entrenamiento modifica los pesos de la red de forma que produzca vectores de salida consistentes. El proceso de entrenamiento extrae las propiedades estadísticas del conjunto de vectores de entrenamiento y agrupa en clases los vectores similares (Basogain Olabe , 2009).

Las técnicas usadas para el aprendizaje y la clasificación pueden organizarse atendiendo a su naturaleza en: métodos estadísticos, modelos o algoritmos matemáticos para el reconocimiento de patrones, estrategias basadas en árboles de decisión y sistemas basados en el conocimiento (SBC) entre otros. En esta tesis son interés de estudio los SBC.

1.1.3 Sistemas basados en el conocimiento de la inteligencia artificial

En (Piñero Pérez, 2005) de forma genérica, la estructura de un Sistema Basado en el Conocimiento (SBC) es comprendida como el compendio de:

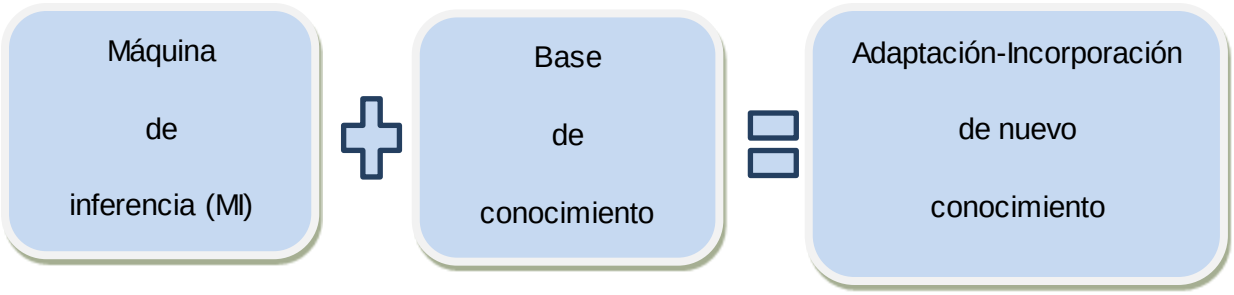


Figura 1. Estructura genérica de un SBC (adaptado de (Piñero Pérez, 2005))

La Base de Conocimiento (BC) es uno de los elementos fundamentales pues constituye la estructura encargada de almacenar el conocimiento. La Máquina de Inferencia (MI) es el método de solución que *"razona"* usando el conocimiento contenido en la BC. Como es apreciado, una de las principales características que distinguen a los SBC es la distinción entre conocimiento y estrategia de control utilizada para el manejo de este.

Numerosos son los SBC diferenciados por la forma de representar el conocimiento y por la naturaleza de la máquina de inferencia asociada, algunos ejemplos de ellos son:

- Sistemas Basados en Probabilidades (SBP)
- Sistemas Basados en Reglas (SBR)
- Sistemas Basados en Casos (CBR)
- Sistemas de Inferencia Difusos (SIB)

Son interés de este trabajo los SIB, por lo que en la siguiente sección se analiza de forma detallada sus fundamentos teóricos y se ofrece un ejemplo de como crearlo y explotarlo.

1.1.4 Sistemas de inferencia difusos

Es un sistema computacional basado en los conceptos de la teoría de los conjuntos difusos cuyos componentes son (Zadeh, 1976):

- Una base de reglas difusas.
- Una base de datos que contiene las funciones de membresía a utilizar.
- Un mecanismo de razonamiento, el cual ejecuta el procedimiento de inferencia.

La base del funcionamiento de los SIB son los conjuntos difusos. En estos conjuntos la pertenencia de un elemento se convierte en un problema de grado. Más formalmente se puede decir que un conjunto difuso ***A*** en un universo de discurso ***U*** está caracterizado por una función

de pertenencia μ_a la cual a cada elemento en el dominio le asigna un grado de pertenencia al conjunto en el intervalo [0,1] y se representa de la forma μ_a : **U**$\rightarrow$[0, 1] (Cox, y otros, 1998) (Zadeh, 1976). De esta forma un mismo elemento puede pertenecer a varios conjuntos simultáneamente solo que con diferente grado de pertenencia. Cada conjunto difuso tiene asociado además un término lingüístico de forma tal que la función de pertenencia asociada a un conjunto está ligada a una palabra como por ejemplo: bajo, medio o alto.

En la representación y construcción de las funciones de pertenencia se pueden utilizar diferentes modelos matemáticos tales como: funciones triangulares, funciones trapezoidales, funciones campana y funciones simoidales (véase Anexo 1).

El centro de las técnicas de modelación difusa lo constituyen las *variables lingüísticas,* concepto que agrupa a los conjuntos difusos asociados a una misma variable. Una *variable lingüística* (Zadeh, 1976) (Herrera, 2001) se caracteriza por un quíntuplo ***(x, T(x), X, G, M)*** en el cual:

x	nombre de la variable
T(X)	conjunto de términos lingüísticos
X	universo de discurso
M	regla semántica[1]
G	conjunto de reglas sintácticas[2]

Las variables lingüísticas y sus conjuntos difusos son empleados para describir relaciones entre variables en forma de reglas *"si-entonces"* que se les conoce como reglas difusas. Se define una regla difusa *R* como una tupla (P, Q) donde *P* son los conjuntos difusos que representan a los antecedentes y *Q* es el consecuente (Cox, y otros, 1998).

---

[1] *M* asocia a cada *valor lingüístico Z* su significado *M(Z)*, donde *M(Z)* denota un conjunto difuso en *X*

[2] Describe la relación entre los conjuntos difusos

Una característica importante que distingue a los SIB de los Sistemas Basados en Reglas clásicos es que el resultado de la inferencia se obtiene de aplicar numerosas reglas (generalmente todas las reglas) y conciliar las inferencias parciales de estas a partir de un procedimiento de agregación (véase Anexo 3).

En general el esquema de evaluación de cada regla se basa en la aplicación de una T-Norma mientras que el procedimiento de agregación se basa en la aplicación de una Co-Norma. Como ejemplos de T-Normas y Co-Normas usualmente se usan las combinaciones min-max y producto-max (Cox, y otros, 1998) (Nauck, y otros, 1999).

Existen tres tipos fundamentales de sistemas de inferencia difusos: el modelo de Mamdani (Mamdani, 1977) así como el Sugeno y Tsukamoto reportados en (Takagi, y otros, 1985) que se diferencian en la forma del consecuente de sus reglas difusas (Cox, y otros, 1998). Para la aplicación en problemas de clasificación, con frecuencia se utiliza el Sugeno grado cero precisamente por la estructura de las reglas que este tipo de sistema representa (Botía-Blaya, 2003). Este modelo también conocido como TSK, por sus autores Takagi, Sugeno y Kang, utiliza reglas de la forma: **Si** "*x es A**",*** mientras "*y es B**"*** **entonces** *z=f(x, y)*, donde *A* y *B* son conjuntos difusos en el antecedente mientras *z* usualmente *f(x, y)* es un polinomio cuyo grado determina el grado del modelo difuso. Cuando *f* es una constante al sistema difuso correspondiente se le llama "*modelo difuso Sugeno de grado cero*". En Mamdani por otra parte el antecedente y el consecuente de las reglas son conjuntos difusos con la peculiaridad que la regla puede tener múltiples antecedentes.

1.1.5 Ejemplo de creación y explotación de un SIB Sugeno grado cero

Con el propósito de comprender la aplicación práctica de los conceptos analizados en las secciones anteriores, se presenta un ejemplo de SIB Sugeno Grado Cero que actualmente se utiliza en GESPRO para evaluar la ejecución de un proyecto en una fecha de corte, el cual se describe formalmente en (Lugo García, y otros, 2012) y tiene como propósito mezclar los resultados de siete indicadores, para obtener una evaluación final del estado del proyecto.

La metodología que se emplea en el diseño del SIB es la siguiente:

- Definir conjuntos difusos para cada indicador.
- Construir funciones de pertenencia para los conjuntos difusos.
- Declarar las reglas difusas.
- Evaluación de las reglas.

Los indicadores para los que deberán construirse los conjuntos difusos y sus variables lingüísticas son los siguientes: IRE, IRP, IRC, ICD, IRL, IRRH e IREF. Para todos los indicadores, los conjuntos difusos definidos son “Bajo”, “Medio” y “Alto” y sus intervalos aparecen definidos en la Tabla .

- Indicador-	- Evaluación -										
	“Bajo”				“Medio”			“Alto”			
	a	*b*	*c*	*d*	*a*	*b*	*c*	*a*	*b*	*c*	*d*
IRE	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-
IRP	-	-	0.9	1	0.9	1	1.1	1	1.1	-	-
IRC	-	-	0.9	1	0.9	1	1.1	1	1.1	-	-
ICD	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-
IRL	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-
IRRH	-	-	0.2	0.5	0.4	0.6	0.8	0.6	0.8	-	-
IREF	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-

Tabla 1. Intervalos definidos para los conjuntos difusos (tomado de (Lugo García, y otros, 2012))

Las funciones de pertenencia de cada conjunto difuso se representan utilizando funciones matemáticas trapezoidales (para “Bajo” y “Alto”) y triangulares (para “Medio”) como se observa en la Figura 2.

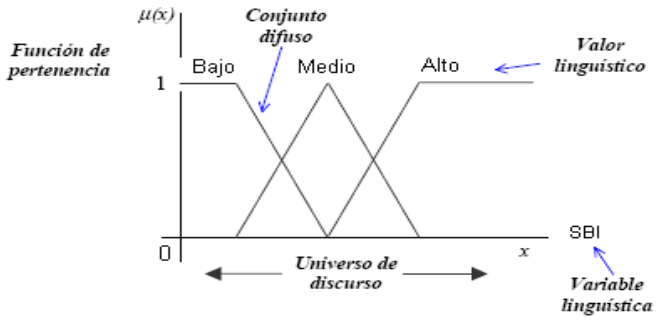


Figura 2. Representación gráfica de los conjuntos difusos (tomado de (Lugo García, y otros, 2012)).

Ajustando las definiciones clásicas de las funciones trapezoidales y triangulares a los conjuntos definidos en la Figura 2 y considerando los intervalos definidos en la Tabla :

La función de pertenencia al conjunto difuso “Bajo” dependería de dos parámetros px1 y px2 y quedaría denotada en (1).

(1)
$$\mu(x) = \begin{cases} 1, & 0 \le x < px1 \\ \frac{(px2 - x)}{(px2 - px1)}, & px1 \le x \le px2 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

La función de pertenencia al conjunto difuso “Medio” dependería de tres parámetros py1, py2, py3 y quedaría denotada en (2).

(2)
$$\mu(x) = \begin{cases} \frac{(x - py1)}{(py2 - py1)}, & py1 \leq x < py2 \\ \frac{(py3 - x)}{(py3 - py2)}, & py2 \leq x \leq py3 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

La función de pertenencia al conjunto difuso “Alto” dependería de dos parámetros pz1 y pz2 y quedaría denotada seguun (3).

(3)
$$\mu(x) = \begin{cases} \frac{(x - pz1)}{(pz2 - pz1)}, & pz1 \leq x < pz2 \\ 1, & x \geq pz2 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

Mediante la priorización de la evaluación de los indicadores, las variables lingüísticas y sus conjuntos difusos son utilizados para escribir las reglas difusas. En el trabajo se crearon un total de 27 reglas difusas para evaluar el proyecto. El enunciado de estas reglas se encuentran en (Piñero Pérez, 2013). Las reglas que se definieron tienen la forma:

Regla 1: IF IND1 is Bajo AND IND2 is Bajo AND …INDn is Bajo, then “Ejecución_Proyecto Mal”;

Regla 2: IF IND1 is Medio AND IND2 is Medio AND … INDn is Medio, then “Ejecución_Proyecto Regular”;

Regla 3: IF IND1 is Alto AND IND2 is Alto AND … INDn is Alto, then “Ejecución_Proyecto Bien”;

Mediante la agregación difusa, las reglas evaluadas producen como única salida los grados de pertenencia a los conjuntos difusos que se corresponden con la evaluación Mal, Regular o Bien alcanzada por el proyecto. Para obtener la evaluación cualitativa final del proyecto se aplica el método Sugeno Grado Cero descrito en la

Figura 3. El modelo de funcionamiento del sistema difuso se basa en los siguientes principios:

- Se utilizará como TNorma, el Producto.
- Se utilizará como CoNorma, el Max.

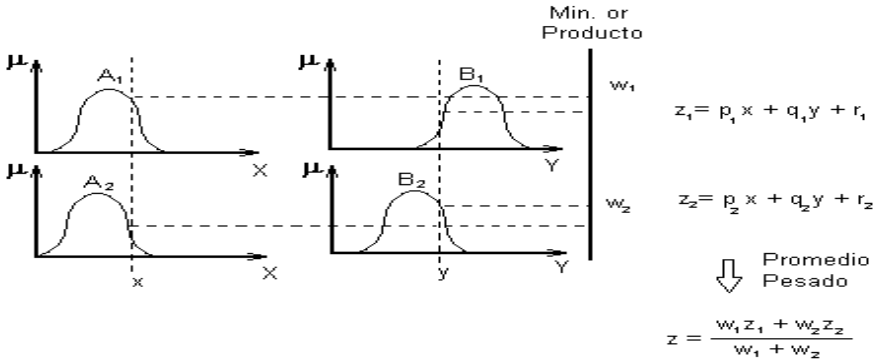
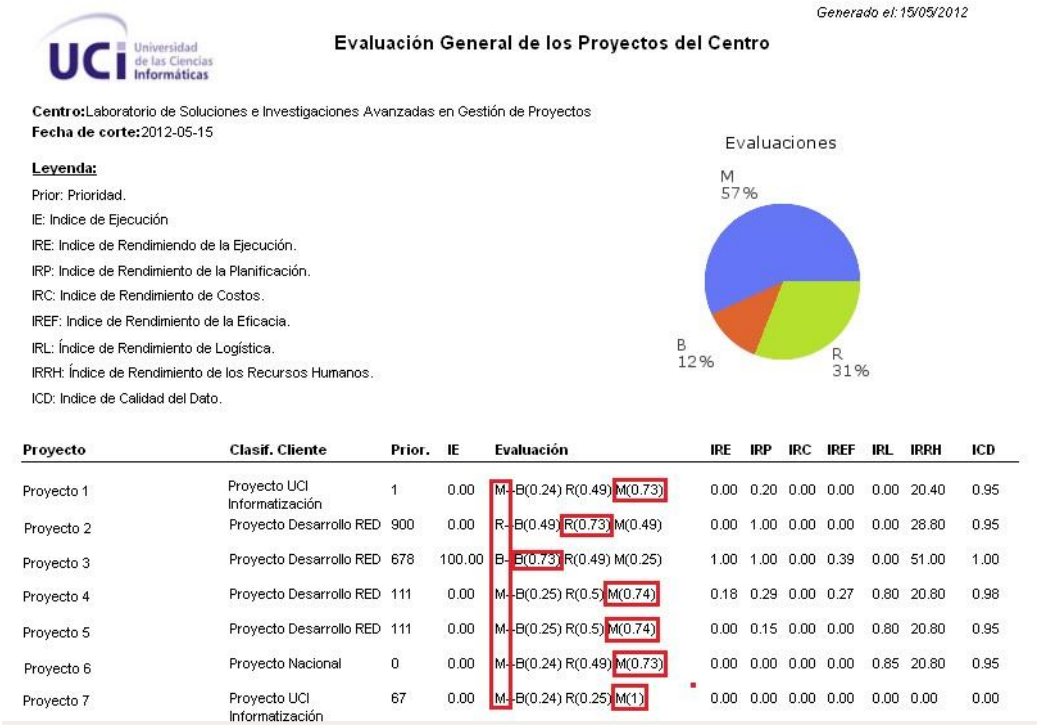


Figura 3. Modelo de funcionamiento del SIB (tomado de (Lugo García, y otros, 2012)).

Este SIB que se ha descrito anteriormente es empleado para generar el Reporte "Evaluación General de los Proyectos del Centro" el cual propone la evaluación de la ejecución Bien (B), Regular (R) o Mal (M) de los proyectos en una fecha de corte, como observa en Figura 4.



Generado el: 15/05/2012

UCI Universidad de las Ciencias Informáticas

Evaluación General de los Proyectos del Centro

Centro: Laboratorio de Soluciones e Investigaciones Avanzadas en Gestión de Proyectos
Fecha de corte: 2012-05-15

Leyenda:
Prior: Prioridad.
IE: Indice de Ejecución
IRE: Indice de Rendimiendo de la Ejecución.
IRP: Indice de Rendimiento de la Planificación.
IRC: Indice de Rendimiento de Costos.
IREF: Indice de Rendimiento de la Eficacia.
IRL: Índice de Rendimiento de Logística.
IRRH: Índice de Rendimiento de los Recursos Humanos.
ICD: Indice de Calidad del Dato.

Proyecto	Clasif. Cliente	Prior.	IE	Evaluación	IRE	IRP	IRC	IREF	IRL	IRRH	ICD
Proyecto 1	Proyecto UCI Informatización	1	0.00	M-B(0.24) R(0.49) M(0.73)	0.00	0.20	0.00	0.00	0.00	20.40	0.95
Proyecto 2	Proyecto Desarrollo RED	900	0.00	R-B(0.49) R(0.73) M(0.49)	0.00	1.00	0.00	0.00	0.00	28.80	0.95
Proyecto 3	Proyecto Desarrollo RED	678	100.00	B-B(0.73) R(0.49) M(0.25)	1.00	1.00	0.00	0.39	0.00	51.00	1.00
Proyecto 4	Proyecto Desarrollo RED	111	0.00	M-B(0.25) R(0.5) M(0.74)	0.18	0.29	0.00	0.27	0.80	20.80	0.98
Proyecto 5	Proyecto Desarrollo RED	111	0.00	M-B(0.25) R(0.5) M(0.74)	0.00	0.15	0.00	0.00	0.80	20.80	0.95
Proyecto 6	Proyecto Nacional	0	0.00	M-B(0.24) R(0.49) M(0.73)	0.00	0.00	0.00	0.00	0.85	20.80	0.95
Proyecto 7	Proyecto UCI Informatización	67	0.00	M-B(0.24) R(0.25) M(1)	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figura 4. Ejemplo de explotación del SIB (tomado de (Lugo García, y otros, 2012)).

1.2 Estrategias de generación de reglas de inferencia difusa

La estructura y el funcionamiento del SIB dependen de la base de reglas que utilice. La base de reglas se puede construir de modo manual o automático. La tendencia de construcción manual

necesita un alto grado de subjetividad por parte de los expertos que construyen las reglas. Estos podrían confundirse en la definición de los parámetros de los conjuntos difusos o en la combinación de estos para la construcción de las reglas. Por otra parte formar todas las posibles combinaciones de conjuntos difusos para construir las reglas es un problema engorroso si tenemos en cuenta la cantidad de posibles bases de reglas a considerar.

Existe una vasta gama de métodos para producir reglas difusas a partir de los datos, que a groso modo según las estrategias empleadas quedan sujetas a tres principales grupos:

- Generar reglas duras y transformar estas reglas en reglas difusas.
- Generar una base de reglas difusas candidatas y luego en un segundo paso seleccionar un subconjunto de ellas teniendo en cuenta la capacidad de las mismas para representar al conjunto de datos de entrenamiento.
- Generar una base de reglas difusas inicial a partir de los datos que se corresponda en número a la base de reglas final y luego someter dicha base de reglas a un proceso de optimización.

1.2.1 Generación de la base de reglas duras y transformación en reglas difusas.

Explota las potencialidades de los algoritmos para la generación de reglas duras (Mitchell, 1997) por medio de sus tres fases bien determinadas:

- Generación de las reglas duras.
- Construcción de los conjuntos difusos partiendo de los datos.
- Transformación de las reglas duras en difusas.

Varios algoritmos que conciben reglas duras basan su funcionamiento en los árboles de decisión, entre los más conocidos están el ID3 (Quinlan, 1986), C4.5 (Quinlan, 1996), GID3 (Xizhao, y otros, 1998) y el FID3 (Janikow, y otros, 1999).

Otra estrategia en la generación de reglas difusas a partir de reglas duras es el uso de las redes neuronales (Setiono, 1998) (Zhou, y otros, 2003) (Setiono, y otros, 2000). Estos algoritmos no son los preferidos para la extracción de reglas, no obstante, experimentos realizados dicen que las reglas extraídas desde redes neuronales son comparables con la extraídas desde los árboles de decisión en términos de cantidad de reglas, exactitud de las predicciones y número promedio de condiciones de la regla.

El uso de método basados en conjuntos aproximados (rough sets) para la generación de reglas duras y su posterior conversión a difusas es otra tendencia en la generación de reglas difusas

(Grzymala-Busse, 1992) (Komorowski, y otros, 1999) (Pal, 1997) (Pal, y otros, 2001) (Wanga, y otros, 2003). La idea básica de esta tendencia se basa en la construcción de reductos y la posterior combinación de los valores de los atributos formando reglas y teniendo en cuenta la relación de estos en los casos del conjunto de entrenamiento (Bello, 2005). La eficiencia en la obtención de las reglas usando estos algoritmos depende en gran medida de la complejidad del procedimiento de construcción de los reductos que con frecuencia es costoso.

1.2.2 Generación de la base de reglas difusas candidatas y refinamiento.

Uno de los algoritmos pioneros en esta estrategia lo constituye el algoritmo propuesto por Wang y Mendel (Wang, y otros, 1992). En una primera fase este algoritmo genera todas las posibles reglas que se obtienen de las combinaciones de los conjuntos difusos estos últimos predefinidos de antemano. En un segundo estado refina la base de reglas inicial seleccionando solamente aquellas reglas que sean representativas del conjunto de entrenamiento.

Hong y Lee (Hong, y otros, 1996) proponen un método basado en el uso de tablas de decisión para derivar automáticamente reglas difusas y funciones de pertenencia a partir de datos numéricos. En este algoritmo se construyen inicialmente un conjunto de funciones de pertenencia que son representadas en una tabla de decisión multidimensional. Luego la tabla de decisión es simplificada por medio de un conjunto de operaciones definidas para actuar sobre sus filas y columnas. Finalmente se generan reglas a partir de cada fila de la tabla.

En (Nauck, 2000) se describen en detalle varios algoritmos para la generación de reglas difusas que siguen la estrategia de las reglas candidatas. Estos algoritmos son utilizados en el sistema NEFCLASS y NEFPROX (Nauck, y otros, 1999). En su primera fase los algoritmos propuestos por Nauck construyen por cada caso un grupo de antecedentes hasta lograr cubrir todos los casos del conjunto de entrenamiento. En su segunda fase, el algoritmo construye una base de reglas candidatas a partir de la combinación de los antecedentes construidos en la fase primera. En la última fase del aprendizaje, el algoritmo recorre el conjunto de casos seleccionando de la base de reglas candidatas las reglas que constituyen la base de reglas final tomando como criterio de selección la capacidad de las reglas de representar los casos del conjunto de entrenamiento. Los algoritmos propuestos por Nauck son costosos, se generan durante la creación de la base de reglas candidatas una gran cantidad de reglas muchas de las cuales carecen incluso de significado lógico.

1.2.3 Generación de reglas difusas a partir de la optimización de la base de reglas.

Supone la construcción previa de una base de reglas y su posterior optimización. Algunos métodos construyen la base de reglas simultáneamente con la optimización pero por lo general este proceso tiene lugar dividido en dos etapas bien marcadas. La base de reglas inicial puede ser construida de forma manual por los expertos o siguiendo alguna de las estrategias explicadas anteriormente. Con frecuencia las bases de reglas pueden ser mejoradas a partir de someterlas a un proceso de optimización, con vistas a mejorar el rendimiento del sistema de inferencia difuso asociado a las mismas durante la etapa de explotación. Las tendencias en este campo basan su funcionamiento en alguna de las siguientes estrategias (Klose, 2003):

- Asignarle pesos a las reglas difusas y ajustar estos de forma que mejore el funcionamiento del sistema de inferencia difuso.
- Ajustar los parámetros de las funciones de pertenencia de los conjuntos difusos.
- Enfoque híbrido donde se ajusten los parámetros de las funciones de pertenencia y la estructura de la base de reglas de forma simultánea.

La estrategia de pesado de reglas se basa en asignarle pesos a las reglas y optimizar estos pesos. Ejemplos de estas estrategias se presentan en (Kosko, 1992) (Zimmermann, 1996). El peso de la regla es usado para modificar a salida final de la regla sin modificar su estructura interna. Existen tres estrategias básicas: adjuntar pesos a las conexiones entre las capas, considerar los pesos ubicados en los enlaces entre la capa de reglas y la capa de consecuentes o modificar la salida final del sistema asignándole pesos a las clases de respuesta.

En el enfoque de modificación de las funciones de pertenencia se han impuesto las siguientes dos estrategias: a) Utilizar métodos basados en el gradiente y reemplazar las funciones utilizadas en el sistema difuso por funciones diferenciables y b) No usar métodos de aprendizaje basados en el gradiente sino otros métodos más apropiados en dependencia del problema en cuestión. Un ejemplo de modelo que emplea la estrategia a) es el ANFIS (Jang, 1993) mientras NEFCOM (Rudolf, 1995) y NECFAC (Fabri, J.A., y otros, 2000) (Fabri, 2002) son una muestra de la estrategias b).

1.2.4 Modelos Híbridos

Además de las tres estrategias antes analizadas, presentadas en (Piñero Pérez, 2005) y citadas y ratificadas en (Pérez Pupo, y otros, 2013), el autor de este trabajo considera la existencia de una cuarta estrategia que consiste en la formación de modelos híbridos. Esta estrategia propone la creación de modelos que combinen el uso de distintas técnicas de softcomputing potenciando las ventajas de cada una.

Modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing

Este modelo presentado en (Piñero Pérez, 2005) posee un primer módulo encargado del aprendizaje de reglas difusas, se presentan los algoritmos MLRul y GenRUL5 para la generación de reglas difusas del tipo Sugeno grado cero. Ambos abordan el proceso de aprendizaje dividido en dos etapas: la construcción de las variables lingüísticas y sus conjuntos difusos y la generación de reglas a partir de los conjuntos difusos construidos.

Durante la construcción de los conjuntos difusos se construyen intervalos discretos a partir de las variables continuas, se ofrecen tres métodos: uno manual por el experto y dos automáticos (discretización y clusterización). Estas opciones automáticas pueden influir negativamente en la interpretabilidad de los conjuntos difusos y las reglas generadas.

En la construcción de las variables lingüísticas caso continuo se sugieren dos métodos. El primero basado en técnicas analíticas, impone como restricciones que las funciones de pertenencia que se construyen tienen que ser simétricas y que el área bajo la curva de las mismas debe poderse calcular por métodos exactos. El segundo método basado en algoritmos genéticos que no impone fuertes restricciones pero no debe utilizarse en caso de que el primero pueda ser usado, muestra como desventaja su alto costo computacional y que no garantiza encontrar el óptimo sino solo soluciones cuasi óptimas. El método empleado por ambos algoritmos requiere que todos los conjuntos difusos adyacentes de una misma variable lingüística tengan el mismo grado de solapamiento.

En la construcción de variables lingüísticas caso simbólico se propone una función de pertenencia basada en la frecuencia de aparición de las etiquetas simbólicas respecto a las clases y define una especie de grado de separabilidad entre las etiquetas. Esta función aunque mejor que los modelos basados en probabilidades o el modelo duro reportados en la bibliografía, es dependiente de la naturaleza del conjunto de entrenamiento.

El algoritmo MLRul para la generación de reglas difusas construye particiones recursivas por reordenamientos del conjunto de entrenamiento y tomando como criterio de ordenamiento la selección del atributo con mayor poder de discriminación entre objetos de clases diferentes, según el criterio de selección de rasgos MLRelevance. Este algoritmo genera una base de reglas candidatas y luego las refina, pero a diferencia de otros algoritmos reportados en la bibliografía que usan esta misma estrategia la base de reglas candidatas que genera es consistente; y por tanto puede ser usada para la clasificación desde su construcción sin

necesidad del refinamiento. Este algoritmo no es dependiente del orden de los casos en el conjunto de entrenamiento.

El criterio de selección MLRelevance establece una competencia entre las variables de forma aislada y no tiene en cuenta la correlación que pueda existir entre ellas. Como ventaja se tiene que siempre puede ser definida a diferencia de otros criterios como la Ganancia Radial de Quinlan y que devuelve valores entre 0 y 1, propiedad siempre deseable.

El algoritmo GenRul5 por su parte basa su funcionamiento en el cálculo de reductos. Las reglas generadas por este algoritmo son dependientes del orden de los casos en el conjunto de entrenamiento. Para el cálculo de los reductos este algoritmo utiliza el criterio de selección de rasgos PRelevance, un método heurístico basado en la teoría de los conjuntos aproximados que tiene en cuenta simultáneamente la relevancia aislada de las variables y la correlación entre diferentes grupos de variables.

Algoritmo GenRul5

Este algoritmo construye reglas difusas a partir de los casos del conjunto de entrenamiento que no han sido cubiertos por las reglas generadas con MLRul. A continuación se mencionan brevemente sus pasos.

Paso I. Determinar conjunto de atributos que constituya un reducto.
Propone calcular un conjunto reducto que permita reducir la dimensionalidad de los datos, facilitando el proceso de aprendizaje.

Paso II. Construir los intervalos discretos a partir de los atributos continuos.
Prevé construir los intervalos discretos para cada atributo en los que se puedan agrupar todos los valores de un atributo continuo. En este paso se sugiere tres posibles variantes: el criterio de expertos, la discretización o la clusterización.

Paso III. Construir las variables lingüísticas para cada uno de los atributos continuos.
Para cada atributo continuo se construirá una variable lingüística formada por tantos conjuntos difusos como intervalos discretos se hayan calculado para el atributo en el paso anterior.

Paso IV. Construir las variables lingüísticas para cada uno de los atributos simbólicos.
De forma similar a como ocurre en el caso numérico cada atributo constituirá una variable lingüística y cada valor del atributo estará representado por un conjunto difuso y su función de pertenencia.

Paso V. Sustituir en todos los casos los valores numéricos por el término lingüístico del conjunto difuso donde alcanza mayor grado de pertenencia y tratar a los casos que tengan valores ausentes.
Cada valor numérico de los casos del conjunto de entrenamiento es sustituido por el término lingüístico más apropiado de acuerdo al principio de máxima pertenencia.

Paso VI. Generar reglas a partir del conjunto de entrenamiento

Modelo para la evaluación de proyectos basado en softcomputing

En (Pérez Pupo, y otros, 2013) se presenta un modelo que combina el aprendizaje automatizado de reglas difusas con otras técnicas de softcomputing para la evaluación de proyectos. Este modelo está formado por cinco módulos: 1) Pre-procesamiento de datos, 2) el aprendizaje y construcción de una base de reglas difusas para la evaluación, 3) la optimización de la base de reglas tomando como base algoritmos inspirados en redes neuronales, 4) la explotación de las reglas en la evaluación de proyectos, 5) aprendizaje automatizado como estrategia para la continuidad en el aprendizaje durante la explotación del sistema.

Para el aprendizaje y construcción de la base de reglas se propone un algoritmo para la generación de reglas difusas del tipo Sugeno grado cero que es una adaptación del MLRul presentado en (Piñero Pérez, 2005). En el nuevo algoritmo se propone construir los intervalos y las variables lingüísticas de la misma forma que en MLRul y GenRul5.

Luego se construye un árbol que inicialmente contendrá en su raíz todo el conjunto de entrenamiento, de donde se selecciona el atributo más relevante. A partir de la equivalencia respecto al atributo se particiona el conjunto creando por cada subconjunto una hoja del árbol que será añadida como hija a la hoja actual; este proceso es repetido mientras en una misma hoja queden casos con consecuentes distintos. Luego de cada hoja del árbol se selecciona el caso holotipo y se construye a partir de él una regla difusa. Para la optimización de las reglas se aplica una estrategia centrada en la modificación de los parámetros de los conjuntos difusos a partir de la aplicación de un algoritmo UMDA.

1.3 Metodologías de desarrollo de software

El desarrollo de software es una tarea compleja y riesgosa que involucra un gran equipo de personas trabajando en común por lo que se hace necesario mantener un estricto control sobre los procesos de manera que se garantice la organización y coordinación de todo el trabajo. Según (Presman, 2010) un proceso de desarrollo de software es un conjunto de actividades

para transformar los requerimientos de un usuario en un software, define quién está haciendo qué, cuándo y cómo alcanzar un determinado objetivo.

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las llamadas metodologías tradicionales y las metodologías ágiles. Las tradicionales enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto lo que brinda la ventaja de poder efectuar un futuro mantenimiento de manera más eficiente, son recomendadas para los proyectos de grandes dimensiones y con grandes equipos de desarrollo. En tanto las metodologías ágiles dan mayor importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueven el trabajo en equipo. Su selección depende de qué producto se desee desarrollar, de las dimensiones que tendrá el mismo, del tiempo que se disponga y del equipo de trabajo, entre otros factores.

1.3.1 Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo de Software (RUP[3]) junto al Lenguaje Unificado de Modelado (UML[4]), constituye la metodología más utilizada para el análisis, implementación y documentación de sistemas de software (Ivar, y otros, 2000).

La metodología RUP está definida por tres aspectos fundamentales:

Procesos dirigidos por casos de usos: Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado, el conjunto de todos los casos de usos constituye el modelo de casos de uso y describe la funcionalidad total del sistema, por lo que son usados para especificar los requisitos de un sistema, para guiar su diseño, implementación y prueba, de este modo los casos de usos no solo inician el proceso sino que le proporcionan un hilo conductor.

Centrado en la arquitectura: La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles a un lado. Esta se construye

[3] **RUP:** Acrónimo en inglés de Rational Unified Process

[4] **UML:** Acrónimo en inglés de Unified Model Language

con los casos de usos más significativos del sistema que representan las funciones claves del sistema en desarrollo.

Iterativo e incremental: Como el desarrollo de un proyecto es un proceso complejo, es práctico dividir el trabajo en partes más pequeñas o mini proyectos, cada mini proyecto es una iteración que resulta en un incremento, las necesidades de los usuarios no pueden definirse completamente al principio, por lo que se refinan en iteraciones sucesivas (Ivar, y otros, 2000).

Como se puede comprobar esta metodología divide el trabajo en muchas etapas durante las cuales se genera una gran cantidad de documentación que es muy útil para proyectos de grandes dimensiones, no siendo así para proyectos pequeños para los que se dispone de poco tiempo y de un equipo de desarrollo pequeño como es el caso del diseño del algoritmo basado en GenRul5 para el aprendizaje automatizado.

1.3.2 SCRUM

Es una metodología ágil para el desarrollo de proyectos que brinda un conjunto de buenas prácticas de trabajo con la capacidad de ofrecer valor al producto final y agilidad en el desarrollo. Sus principios son válidos para entornos que trabajan con requisitos inestables y necesitan agilidad. En esta metodología la gestión no se basa en el seguimiento de un plan sino en la adaptación continua a las circunstancias de la evolución del proyecto, está adaptado a las personas antes que a los procesos y emplea un desarrollo ágil, iterativo e incremental.

Características de los campos de SCRUM:

Incertidumbre: Como elemento consustancial y asumido en el entorno y en la cultura de la organización.

Auto-organización: No hay roles de gestión que marquen pautas o asignación de tareas.

Fases de desarrollo solapadas: Los trabajos que se llevan a cabo pierden el carácter de fase y son actividades que se realizan en cualquier momento, de forma simultánea, o a demanda según las necesidades en cada iteración.

Control sutil: Se establecen controles para evitar que el ambiente de ambigüedad, inestabilidad y tensión en el que el equipo trabaja derive hacia el descontrol.

Difusión y transferencia del conocimiento: Todos los miembros del equipo aportan y aprenden del resto del equipo.

Para SCRUM los documentos son soporte de documentación, permiten la transferencia del conocimiento, registran información histórica, son obligatorios en cuestiones legales o normativas, pero son menos importantes que los productos que funcionan, por lo que no pueden sustituir, ni pueden ofrecer la riqueza y generación de valor que se logra con la comunicación directa entre las personas y a través de la interacción con los prototipos. Por eso, siempre que sea posible debe preferirse, y reducir al mínimo indispensable el uso de documentación, que genera trabajo que no aporta un valor directo al producto (Palacio, 2007).

Las características vistas hasta ahora se adecuan a las necesarias para desarrollar un sistema como el que se propone, pero se debe anotar como desventaja que el equipo de desarrollo en cuestión está compuesto solo por dos personas, por lo que se hace necesario minimizar las tareas para lograr un mejor aprovechamiento del tiempo.

1.3.3 eXtreme Programming (XP)

La metodología XP fue concebida y desarrollada para direccionar las necesidades específicas del desarrollo de software llevado a cabo por pequeños equipos en aras de satisfacer requisitos vagos y cambiantes (Beck, 1999).

Los valores originales que la programación extrema fomenta son: simplicidad, comunicación, retroalimentación y coraje. Un quinto valor, respeto, fue añadido en la segunda edición de *Extreme Programming Explained* (Beck , y otros, 2004).

A continuación la Figura 5 muestra las características fundamentales de la metodología expuestas en (Beck, 1999) y ratificadas en (Beck , y otros, 2004):

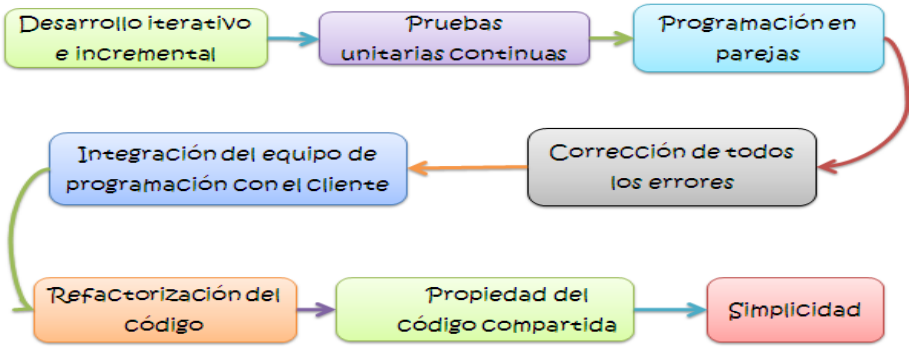


Figura 5. Características fundamentales de la metodología (tomado de (Academics, 2014))

Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.

Pruebas unitarias continuas, frecuentemente repetidas y automatizadas. Se aconseja escribir el código de la prueba antes de la codificación.

Programación en parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Posibilitando del código revisado y discutido mientras se escribe.

Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

Corrección de todos los errores antes de añadir nueva funcionalidad, en consecuencia con ello hacer entregas frecuentes.

Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre éste, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Son fundamentos de esta metodología ágil:

- Escribir pruebas unitarias basadas en los principales procesos lo que permite predecir posibles fallas futuras.
- Integrar y probar el sistema en su conjunto varias veces al día.
- La producción de todos los programas de dos en dos, dos programadores una pantalla.
- Proyectos a partir de un diseño simple que evoluciona constantemente para aumentar la flexibilidad necesaria y eliminar la complejidad innecesaria.
- La reutilización de código para lo cual se crean patrones o modelos estándares, siendo más flexibles al cambio.

En Figura 6 se desglosan cada una de las actividades propuestas por la metodología en cada una de las fases del desarrollo ágil.

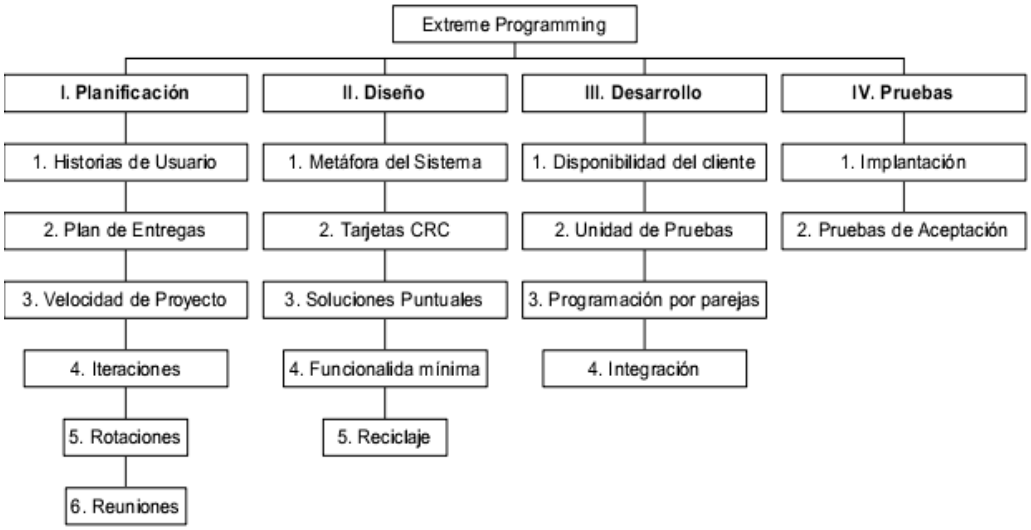


Figura 6. Fases de la metodología XP (tomado de (Fernández Escribano, 2002)

Esta metodología brinda como ventaja un mayor aprovechamiento del tiempo, lo que permite agilizar todo el proceso, pues el tiempo que se invierte en documentar todo el ciclo de desarrollo de un proyecto se aprovecha en la implementación del mismo, pero esto trae consigo implícito una desventaja, si el proyecto es muy abarcador y no se documenta adecuadamente en un futuro cuando se desee dar mantenimiento al mismo puede resultar muy engorroso o casi imposible si no se encuentran presentes los integrantes del equipo de desarrollo original.

1.3.4 Fundamentación de la metodología a utilizar

Después de realizado el estudio de algunas metodologías de desarrollo de software, se selecciona para el desarrollo de la herramienta propuesta XP por ser una metodología ágil, diseñada para equipos de trabajo pequeños, centrada en vincular al cliente en el ciclo de desarrollo, incrementando la posibilidad de éxito, minimizando los riesgos de no conformidades y de obtener un producto final rechazado por el cliente por no cumplir con los objetivos y especificaciones trazadas. La metodología XP responde de manera eficiente a los cambios que se puedan presentar durante todo el desarrollo de la herramienta, proponiendo un ciclo de vida dinámico. El proceso de prueba de XP posibilita probar cada funcionalidad al finalizar cada iteración comprobando si cumple con los requisitos de la herramienta.

1.4 Tecnologías a utilizar en el modelado

1.4.1 Lenguaje Unificado de Modelado UML 2.0

Los lenguajes de modelado son una guía estándar que proveen un conjunto de diagramas, símbolos y mecanismos para diseñar sistemas informáticos. UML es uno de los lenguajes más conocidos y utilizado en la actualidad.

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, pero no especifica en sí mismo qué metodología o proceso utilizar (Hamilton, y otros, 2006).

Algunas de las características representativas de UML son:

- UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso.
- Es orientado a objetos, permitiendo al programador que organice su programa de acuerdo con abstracciones de alto nivel, siendo estas cercanas a la forma de pensar de las personas

1.4.2 Visual Paradigm 8.0

Visual Paradigm es una herramienta que facilita el proceso de desarrollo de software a través del modelado de los artefactos del sistema. Aplicación informática multiplataforma que brinda soporte para todos los diagramas que propone el lenguaje de modelado UML. La misma propicia un conjunto de ayudas para el desarrollo, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Paradigm, 2013). Se caracteriza por:

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad y uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- El modelo y el código permanecen sincronizados en todo el ciclo de desarrollo.
- Generación de bases de datos: Transformación de diagramas de Entidad-Relación en tablas de base de datos.

1.5 Tecnologías a utilizar en el desarrollo

1.5.1 Plataforma y lenguaje de programación R.

En (KABACOFF, 2012) se define a R como una de las plataformas más populares para el análisis y la visualización de datos actualmente disponible. Es un software de código abierto con versiones para los sistemas operativos Windows, Mac OS y Linux. A continuación se enuncian algunas características de la plataforma:

- Implementa prácticamente todas las técnicas necesarias en cualquier escenario de investigación.
- Permite la incorporación de los investigadores a su desarrollo o al desarrollo de módulos específicos para cubrir sus necesidades.
- Es el software estadístico con licencia GNU más extendido a nivel mundial tanto para docencia como para investigación (KDnuggets, 2012) (véase Anexo 7).

Al hacer uso de una base de datos en el proceso de minería de datos llevado a cabo por el algoritmo se precisa utilizar *plr* como lenguaje procedural de R capaz de interactuar con el servidor de base de datos PostgresSQL 9.1.

1.5.2 Sistema Gestor de Bases de Datos (SGBD) PostgreSQL 9.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, cuyo código fuente está disponible libremente. Es uno de los sistemas de gestión de bases de datos de código abierto más potente del mercado.

Las principales características de este SGBD son:

Atomicidad (Indivisible): asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

Aislamiento: una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.

Durabilidad: establece que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema (PostgreSQL, 2010).

1.5.3 Herramienta para la administración de la bases de datos PgAdmin III

Para administrar la base de datos se utiliza la herramienta de diseño y manejo de bases de datos PgAdmin III, con la cual actualmente el equipo de desarrollo de GESPRO trabaja teniendo en cuenta que:

•Está diseñado para responder a las necesidades de diferentes tipos de usuarios, desde la escritura de simples consultas SQL hasta la elaboración de bases de datos complejas.

•La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración.

•No se requieren controladores adicionales para comunicarse con la base de datos del servidor.

•Se encuentra disponible para Windows, Linux, FreeBSD, Solaris y MacOS (Dataprix, 2013).

1.6 Conclusiones del capítulo

Después de realizado el estudio bibliográfico se puede arribar a las siguientes conclusiones:

- Existen cuatro estrategias fundamentales para el aprendizaje de reglas difusas. Las más recientes sugieren la construcción de modelos híbridos que combinen el uso de distintas técnicas de softcomputing potenciando las ventajas de cada una. Ninguna de estas técnicas es ampliamente usada para la toma de decisiones en la gestión de proyectos.

- Se adopta como estrategia de generación de reglas difusas la que propone la creación de métodos híbridos y dentro de ella se selecciona como algoritmo a implementar el GenRul5 propuesto por el Dr. Pedro Piñero en su tesis de doctorado *"Modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing"*.

Capítulo 2: PROPUESTA DE SOLUCIÓN

En el presente capítulo se describe el algoritmo GenRul5 así como la versión del algoritmo a desarrollar. Además se hace una descripción del proceso de desarrollo propuesto en la metodología XP adaptado a la investigación. Se generan todos los artefactos correspondientes a cada fase como es el caso de las historias de usuario y plan de iteraciones en la fase de planificación, las tarjetas clase responsabilidad colaborador (CRC) en la fase de diseño y finalmente en la fase de codificación se generan las tareas de programación y se obtiene el código funcional de la herramienta.

2.1 Descripción de cada paso del algoritmo GenRul5 y las posibles variantes para su correcta ejecución.

El primer paso consiste en determinar el conjunto de atributos que constituya un reducto empleando el criterio de selección de rasgos PRelevance. El criterio PRelevance es una combinación lineal entre la relevancia del atributo R(a) y una función heurística H(A). El cálculo de R(a) es posible hacerlo siguiendo el criterio MLRelevance (López de Mántaras, 1991), la ganancia de Quinlan (Quinlan, 2002), la ganancia radial (López de Mántaras, 1991) o el criterio de López de Mántaras (Larrañaga, y otros, 2002). En este trabajo R(a) se calculará según el criterio MLRelevance.

La construcción de los intervalos discretos para cada atributo constituye el segundo paso del algoritmo. Es posible realizarla mediante una de las tres variantes que a continuación se exponen. Que los intervalos sean propuestos por los especialistas en la temática específica del problema en cuestión. Una segunda opción sería seguir un proceso sencillo de discretización de los atributos (Ruiz Shulcloper, y otros, 2000), tomando como estrategia: la construcción de intervalos de igual tamaño, o de intervalos con igual frecuencia. Por último, una tercera variante se basa en llevar a cabo un proceso de clusterización a nivel de atributo por medio del cual se construyan los intervalos. En este paso del algoritmo se emplearán intervalos propuestos por especialistas en la temática específica del problema en cuestión.

La construcción de las variables lingüísticas para cada uno de los atributos continuos es el tercer paso del algoritmo. Donde para cada atributo se construirá una variable lingüística formada por tantos conjuntos difusos como intervalos discretos se hayan calculado para el atributo en el paso anterior. Es posible el empleo de técnicas de análisis matemático o técnicas

basadas en los algoritmos genéticos en la construcción de las funciones de pertenencia para cada conjunto.

En el cuarto paso se construyen las variables lingüísticas para cada uno de los atributos simbólicos así como las funciones de pertenencia correspondientes a cada uno de los conjuntos difusos. De forma similar a como ocurre en el caso numérico, cada atributo constituirá una variable lingüística y cada valor del atributo estará representado por un conjunto difuso y su respectiva función de pertenencia. Por la naturaleza de la base de datos de proyectos terminados del laboratorio de desarrollo de GESPRO no se hizo necesaria la aplicación de dicho paso al no existir atributos simbólicos.

Seguidamente se sustituye en todos los casos los valores numéricos por el término lingüístico del conjunto difuso donde alcanza mayor grado de pertenencia y tratar a los casos que tengan valores ausentes. En este paso cada valor numérico de los casos del conjunto de entrenamiento es sustituido por el término lingüístico más apropiado de acuerdo al principio de máxima pertenencia (Gutiérrez-Martínez, 2003). También se tratan los casos que tengan valores ausentes con una de las dos estrategias siguientes: ignorar las tuplas con valores ausentes o sustituir el valor ausente por un nuevo término lingüístico llamado *unknown* (Piñero, y otros, 2003). Según las características de la base de datos de proyectos terminados del laboratorio de desarrollo de GESPRO no existen casos que contengan valores ausentes.

Finalmente la generación de reglas a partir del conjunto de entrenamiento, siendo este paso el corazón del funcionamiento del algoritmo GenRul5.

A continuación en Figura 7 se muestra un esquema gráfico para una mejor comprensión de lo antes expuesto.

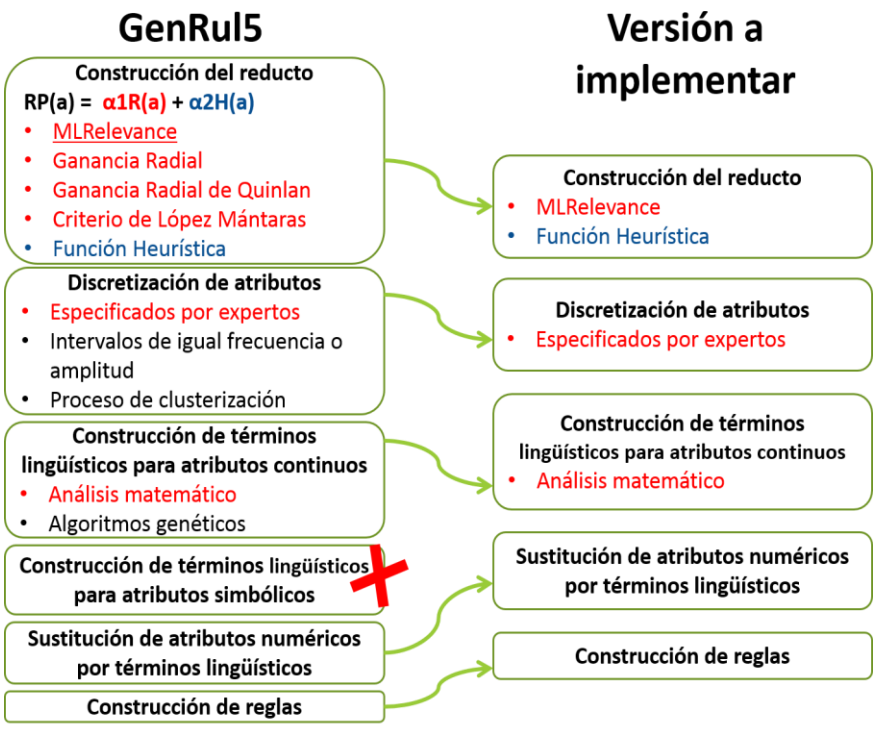


Figura 7. Correspondencia entre los pasos del algoritmo GenRul5 y la variante a desarrollar (del autor)

2.2 Análisis detallado de la versión del algoritmo GenRul5 empleado en la solución.

Antes de comenzar el análisis de los pasos se precisa responder la interrogante ¿Qué es un reducto? Un reducto es un conjunto mínimo de rasgos que preserva una partición del universo. Los métodos de selección de rasgos incluyen dos componentes, una función de evaluación de los subconjuntos y un procedimiento para la generación de subconjuntos. Como el problema de la selección de rasgos es de alto costo computacional, dado N rasgos hay 2N-1 subconjuntos posibles, usualmente el procedimiento de generación de subconjuntos se basa en un método de búsqueda heurística. Existen diferentes medidas para realizar la evaluación de subconjuntos, entre ellas está la medida de calidad de la clasificación. Construir un reducto no tiene un alto costo computacional, pero tratar de encontrar un reducto con la cantidad menor de rasgos posible si tiene un alto costo (Bello, y otros, 2010).

2.2.1 Discusión del Paso I. Determinar conjunto de atributos que constituya un reducto.

Este paso del algoritmo prevé calcular un conjunto reducto que permita reducir la dimensionalidad de los datos, facilitando el proceso de aprendizaje. En general el cálculo de

reductos es un proceso altamente costoso por eso no es recomendable incluir la mayoría de los algoritmos reportados para la construcción de reductos como métodos empotrados dentro de algoritmos de aprendizaje. Sin embargo, en los últimos años se han desarrollado nuevos algoritmos de cálculo de reductos que permiten la determinación de los mismos en tiempos aceptables. Algunas estrategias en este sentido lo constituyen los algoritmos LEX y RSReduct (Caballero, 2004) en su variante de la heurística 2 (véase Anexo 5).

En general se puede decir que es un algoritmo Greedy que comienza por un conjunto vacío de atributos y a través de heurísticas construye un buen reducto. La heurística que se plantea es una combinación lineal de la Ganancia Radial de Quinlan y la heurística H(A) que forma parte del criterio de selección de rasgos PRelevance (Pawlak, 1991) que se presenta en la siguiente subsección de esta tesis.

Criterio de selección PRelevance basado en la heurística H(A)

El criterio PRelevance es una combinación lineal del criterio MLRelevance y una función heurística. El cálculo de la heurística utilizada en PRelevance está basado en la teoría de los conjuntos aproximados (Komorowski, y otros, 1999). Para una mejor comprensión del algoritmo es necesario que se recuerden los conceptos de aproximación superior, aproximación inferior, región positiva, región negativa y región frontera (Komorowski, y otros, 1999) (véase Anexo 6). A continuación se recuerda la definición de dependencia en grado *k* (Komorowski, y otros, 1999) que es el centro del cálculo de la dependencia entre dos conjuntos de atributos en que se basa el criterio PRelevance.

Dependencia en grado k: Se dice que dados dos conjuntos de atributos ***B*** y ***D***, el conjunto de atributos ***D*** depende del conjunto de atributos ***B*** en grado k si existe un valor $k \in [0,1]$ resultante de evaluar en (4). Si $k=1$ se dice además que ***D*** depende totalmente de ***B***, mientras que si $k<1$ se dice que ***D*** depende parcialmente de ***B***.

(4) $$k(B,D) = \frac{\left|POS_B(D)\right|}{\left|U\right|} \quad \text{donde} \quad POS_B(D) = \bigcup_{X \in U/D} \underline{B}X$$

La expresión $\underline{B}X$ denota la aproximación inferior del conjunto X respecto al conjunto de atributos B (Quinlan, 1986).

Finalmente en (5) se representa la expresión que permite calcular el grado de relevancia de un atributo ***a*** según el criterio PRelevance. Como se ve la relevancia de cada atributo es una combinación lineal de su relevancia, tomando el atributo de forma independiente, y de la relevancia grupal del conjunto B de atributos tal que a$\in$ B. El atributo que maximice RP(a) es considerado el más relevante.

(5) $RP(a) = \alpha 1 R(a) + \alpha 2 H(a)$ Tal que $\alpha 1, \alpha 2 \in [0, 1]$, $\alpha 1 + \alpha 2 = 1$

El cálculo de R(a) se realizará siguiendo el criterio MLRelevance teniendo en cuenta los resultados de las tablas comparativas como parte del proceso de validación de resultados en (Piñero Pérez, 2005). A continuación se detalla el criterio MLRelevance.

Criterio de selección de rasgos MLRelevance

En esta sección se adopta para la selección de rasgos el criterio MLRelevance (López de Mántaras, 1991). Suponga un atributo (***a***) con *i* = 1,2*,…,k* valores admisibles, *S* un conjunto de casos y S_i el subconjunto de *S* que contiene los casos que tiene el valor *i* en el atributo ***a***. Entonces la expresión $\frac{|S_i|}{|S|}$ es la frecuencia relativa de aparición del valor *i* en *S.* En (6) se muestra como calcular la relevancia del atributo ***a*** usando el criterio MLRelevance, donde *R*(*a*) es la medida de relevancia del atributo ***a*** en el conjunto *S*, *k* es el número de valores diferentes del atributo ***a*** y C_i es el número de clases diferentes presentes en objetos de *S* que tienen el valor *i* en el atributo ***a***.

(6) $$R(a) = \sum_{i=1}^{K} \frac{|S_i|}{|S|} e^{(1-C_i)}$$

El cálculo de H(a) se propone realizar según el siguiente algoritmo.

Algoritmo 1 para el cálculo de H(a*)*

P1- Calcular el vector R(*T*) = ($R(a_1), R(a_2), R(a_3), \ldots, R(a_{r(a)})$) con $T \subseteq A$ donde *A* es el conjunto de todos los posibles atributos y *T* subconjunto de *A* formado por todos los atributos que pueden ser seleccionados como el más relevante en un momento dado. En caso que las condiciones del problema dicten lo contrario el conjunto *A* estará formado por todos los atributos presentes en el conjunto de entrenamiento.

P2- Determinar los N mejores atributos, siendo los mejores aquellos que maximizan $R(a_i)$. Como resultado de este paso obtenemos el vector, RA = ($R(a_j)$, $R(a_j)$, …, $R(a_t)$) tal que N = |RA|.

P3- Formar todas las combinaciones de ***N*** en ***p*** con los atributos seleccionados en el paso anterior. Como resultado de este paso obtenemos el conjunto formado por la unión de todas las posibles combinaciones. $\mathrm{Comb} = (\{a_i, a_j, a_k\}, \ldots \{a_i, a_t\ a_p\})$

Por ejemplo si tenemos que N = 4, p = 3 y suponiendo que los atributos seleccionados en el paso anterior fueron (a_1, a_3, a_5, a_8) el conjunto combinación tiene $C_p^n = \frac{n!}{p!(n-p)!} = 4$ elementos que son las siguientes combinaciones $\mathrm{Comb} = (\{a_1, a_3, a_5\}, \{a_1, a_3, a_8\}, \{a_3, a_5, a_8\}, \{a_1, a_5, a_8\})$.

P4- Calcular el grado de dependencia de las clases con respecto a cada una de las combinaciones formadas en el paso anterior utilizando la ecuación (4). Como resultado de este paso obtenemos el vector de dependencias *DEP = (k(Comb$_1$, d), k(Comb$_2$, d), … k(Comb$_n$,d))*.

P5- Para cada atributo ***a*** calculamos el valor de H(a) según (7).

(7) $$H(a) = \frac{\sum_{\forall i / a \in Comb_i} k(Comb_i, d)}{\sum_{\forall i} k(Comb_i, d)}$$

Luego de haber aplicado la expresión (5) y determinar el atributo que maximice *RP(a)* se analizan todas las combinaciones obtenidas en el paso P3 donde se encuentre presente el atributo ***a***. Posteriormente a partir de los valores calculados en el vector de dependencias de la clase respecto a las combinaciones se selecciona la combinación que posea mayor dependencia.

2.2.2 Discusión del Paso II: Construir los intervalos discretos a partir de los atributos continuos.

En este paso del algoritmo se convierten los atributos continuos en discretos, o sea construir los intervalos discretos para cada atributo en los que se puedan agrupar todos los valores de un atributo continuo.

En este paso se emplean intervalos propuestos por especialistas en la temática específica del problema en cuestión. Esta vía debe utilizarse cuando los especialistas sean capaces de describir estos intervalos tomando como base su experiencia previa. A continuación los intervalos propuestos de forma manual por especialistas en (Lugo García, y otros, 2012).

Indicadores	Bajo				Medio			Alto			
	a	b	c	d	a	b	c	a	b	c	d
IRE	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-
IRP	-	-	0.9	1	0.9	1	1.1	1	1.1	-	-
ICD	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-
IRRH	-	-	0.2	0.5	0.4	0.6	0.8	0.6	0.8	-	-
IREF	-	-	0.5	0.7	0.5	0.7	0.9	0.7	0.9	-	-

Tabla 2. Conjuntos difusos definidos para cada indicador (adaptado de (Lugo García, y otros, 2012))

Al finalizar este paso se conoce de cada atributo procesado como se pueden agrupar sus valores en intervalos discretos.

2.2.3 Discusión del Paso III: Construir las variables lingüísticas para cada uno de los atributos continuos.

Para cada atributo continuo se construirá una variable lingüística formada por tantos conjuntos difusos como intervalos discretos se hayan determinado para el atributo en el paso anterior.

Básicamente, se ha definido el trabajo con funciones triangulares y trapezoidales como se muestra en (1), (2) y (3). También pueden ser utilizadas funciones de pertenencia campanas gaussianas, curvas beta u otros tipos (véase Anexo 1).

A continuación se muestra el comportamiento de las variables lingüísticas correspondientes a los indicadores IRE, ICD e IREF.

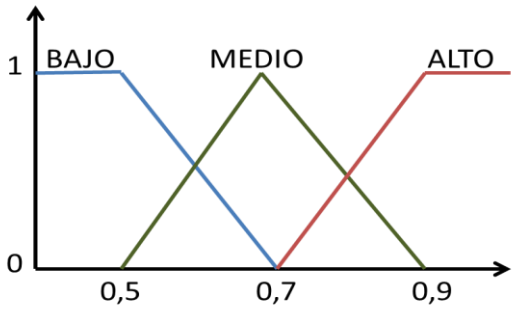


Figura 9. Conjuntos difusos de los indicadores IRE, ICD e IREF (del autor)

La función de pertenencia correspondiente al conjunto difuso *"BAJO"* de los tres indicadores antes mencionados se muestra en (8).

$$(8) \qquad \mu(x) = \begin{cases} 1 & 0 \leq x < 0{,}5 \\ \frac{0{,}7 - x}{0{,}7 - 0{,}5} & 0{,}5 \leq x \leq 0{,}7 \\ 0 & \textit{En cualquier otro caso} \end{cases}$$

La función de pertenencia correspondiente al conjunto difuso *"MEDIO"* de los tres indicadores antes mencionados se muestra en (9).

$$(9) \qquad \mu(x) = \begin{cases} \frac{x - 0{,}5}{0{,}7 - 0{,}5} & 0{,}5 \leq x < 0{,}7 \\ \frac{0{,}9 - x}{0{,}9 - 0{,}7} & 0{,}7 \leq x \leq 0{,}9 \\ 0 & \textit{En cualquier otro caso} \end{cases}$$

La función de pertenencia correspondiente al conjunto difuso *"ALTO"* de los tres indicadores antes mencionados se muestra en (10).

$$(10) \qquad \mu(x) = \begin{cases} \frac{x - 0{,}7}{0{,}9 - 0{,}7} & 0{,}7 \leq x < 0{,}9 \\ 1 & x \geq 0{,}9 \\ 0 & \textit{En cualquier otro caso} \end{cases}$$

En la siguiente gráfica de muestra el comportamiento de las variables lingüísticas correspondiente al indicador IRP.

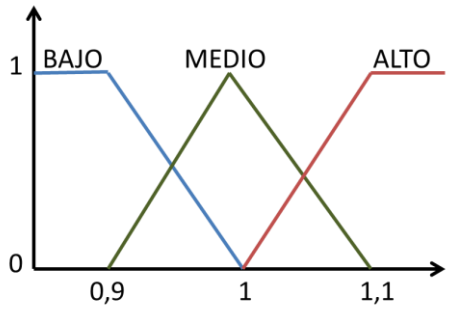


Figura 10. Conjuntos difusos del indicador IRP (del autor)

Se muestra la función de pertenencia correspondiente al conjunto difuso *"BAJO"* en (11).

$$(11) \quad \mu(x) = \begin{cases} 1 & 0 \leq x < 0{,}9 \\ \frac{1-x}{1-0{,}9} & 0{,}9 \leq x \leq 1 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

Se muestra la función de pertenencia correspondiente al conjunto difuso *"MEDIO"* en (12).

$$(12) \quad \mu(x) = \begin{cases} \frac{x-0{,}9}{1-0{,}9} & 0{,}9 \leq x < 1 \\ \frac{1{,}1-x}{1{,}1-1} & 1 \leq x \leq 1{,}1 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

Se muestra la función de pertenencia correspondiente al conjunto difuso *"ALTO"* en (13).

$$(13) \quad \mu(x) = \begin{cases} \frac{x-1}{1{,}1-1} & 1 \leq x < 1{,}1 \\ 1 & x \geq 1{,}1 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

En la siguiente gráfica de muestra el comportamiento de las variables lingüísticas correspondiente al indicador IRRH.

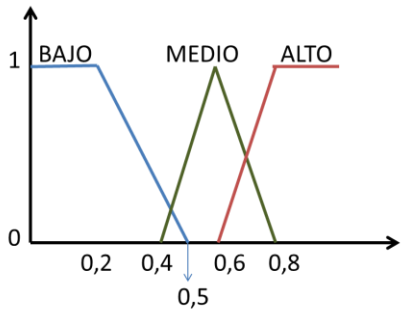


Figura 11. Conjuntos difusos del indicador IRRH (del autor)

Se muestra la función de pertenencia correspondiente al conjunto difuso *"BAJO"* en (14).

$$(14) \quad \mu(x) = \begin{cases} 1 & 0 \leq x < 0{,}9 \\ \frac{1-x}{1-0{,}9} & 0{,}9 \leq x \leq 1 \\ 0 & En\ cualquier\ otro\ caso \end{cases}$$

Se muestra la función de pertenencia correspondiente al conjunto difuso *"MEDIO"* en (15).

$$(15)\quad \mu(x) = \begin{cases} \frac{x - 0{,}9}{1 - 0{,}9} & 0{,}9 \leq x < 1 \\ \frac{1{,}1 - x}{1{,}1 - 1} & 1 \leq x \leq 1{,}1 \\ 0 & \textit{En cualquier otro caso} \end{cases}$$

Se muestra la función de pertenencia correspondiente al conjunto difuso *"ALTO"* en (16).

$$(16)\quad \mu(x) = \begin{cases} \frac{x - 1}{1{,}1 - 1} & 1 \leq x < 1{,}1 \\ 1 & x \geq 1{,}1 \\ 0 & \textit{En cualquier otro caso} \end{cases}$$

2.2.4 Discusión del Paso IV: Sustituir los valores numéricos por el término lingüístico

En este paso cada valor numérico de los casos del conjunto de entrenamiento es sustituido por el término lingüístico más apropiado de acuerdo al principio de máxima pertenencia (Gutiérrez-Martínez, 2003).

Cada caso que contenga un valor en alguna de sus variables que alcance el máximo grado de pertenencia para más de un conjunto difuso se replicará, o sea se generaran dos nuevos casos simbólicos cada uno conteniendo uno de los términos lingüísticos correspondientes a los conjuntos difusos para los que alcanzó la máxima pertenencia como se muestra a continuación.

Dado el caso:

Casos	**IRE**	**IRP**	**ICD**	**IRRH**	**IREF**	**Clase**
Caso 1	Medio **(0.5)** Bajo **(0.5)**	Alto	Medio	Bajo	Bajo	Regular

Se generan los siguientes casos:

Nuevo_Caso 1.1	Medio **(0.5)**	Alto	Medio	Bajo	Bajo	Regular
Nuevo_Caso 1.2	Bajo **(0.5)**	Alto	Medio	Bajo	Bajo	Regular

2.2.5 Discusión del Paso V: Generar reglas a partir del conjunto de entrenamiento

Este paso es el corazón del funcionamiento del algoritmo GenRul5 y se describe en la siguiente definición que es la base del algoritmo propuesto.

Definición 1

Se define que un caso *w* es cubierto por una regla *R = (P, Q)* con antecedentes *P* y consecuente *Q* en grado *k* y respecto a un conjunto de atributos x^w si y solo si $\forall x^w_i \in x^w \exists P_j \in P$ tal que $k \leq \min_{\forall i} (\text{evaluar}(x^w_i, P_j))$ $k \in (0,1]$ y además que *Q* es igual a la clase que corresponde el caso *w*. O sea que al evaluar el conjunto difuso asociado a P_j en el valor $x^w_i \forall$ i se obtiene una pertenencia al conjunto mayor que *k*.

Algoritmo 1: Generar la base de reglas según estrategia de cubrimiento de casos

1. Crear la base de reglas difusas R, inicialmente R = φ
2. Si todos los casos del conjunto de entrenamiento están marcados como visitados entonces terminar en caso contrario ir al paso 3
3. Seleccionar un caso del conjunto de entrenamiento **w** que no esté marcado
4. Si el caso **w** seleccionado no está cubierto por alguna de las reglas (ver Definición 1) de la base de reglas R entonces generar la regla asociada al mismo. Se propone tomar como criterio para el cubrimiento un k > 0.8. En la generación de la regla se utiliza el reducto construido anteriormente, como el conjunto de atributos $\mathbf{x^w}$, el caso **w** y aplicamos el Algoritmo 2.
5. Marcar el caso **w** como visitado e ir al paso P2

Algoritmo 2: Generación de una regla difusa a partir de un caso w y atributos $\mathbf{x^w}$

- A partir de cada atributo del caso w_i que pertenezca a x^w, construir un antecedente de la regla que contenga el conjunto difuso cuyo término lingüístico se corresponda con el valor de este atributo.
- En caso que el valor del atributo sea el término *unknown* ignorar este atributo en la generación de la regla.
- Formar una regla por la combinación de los antecedentes construidos tomando como consecuente de la regla la clase del caso.

2.3 Objeto de informatización

Con el presente trabajo se pretende obtener una serie de funciones que serán integradas al GESPRO como una extensión de este. Mediante la informatización del proceso de generación de reglas de inferencia difusa se ayudará a los especialistas del área de la Gestión de Proyectos en la toma de decisiones.

2.4 Fase de planificación

Para el desarrollo de la herramienta se dividirá el trabajo en cuatro fases como propone la metodología Programación Extrema.

En la fase de planificación se lleva a cabo el proceso enmarcado dentro de la ingeniería de requerimientos, se crean las historias de usuario para obtener una descripción de las funcionalidades con que debe cumplir el sistema, se listan los requerimientos del mismo para tener una mejor visión y finalmente se planifica el tiempo que durará el desarrollo mediante un plan de iteración.

2.4.1 Historias de usuarios

Durante la fase de planificación se generan las historias de usuario, que son los documentos de especificación funcional de una aplicación, son escritas por el cliente en su propio lenguaje con descripciones cortas de que debe hacer el sistema. A continuación se describen dos de las principales historias de usuario del sistema a desarrollar de un total de cinco, el resto se encuentra en los anexos del trabajo (véase Anexo 10):

Historia de Usuario	
Código:HU_#_1	**Nombre Historia de Usuario:** Determinar conjunto de atributos que constituya un reducto
Referencia: RF_#_1, RF_#_2, RF_#_3, RF_#_4, RF_#_5, RF_#_6, RF_#_7, RF_#_8, RF_#_9	
Programador responsable: Eden Sobrino Quiñones	Iteración: 1
Prioridad: Muy Alta	Puntos Estimados: 7 días
Riesgo en Desarrollo: Alta	Puntos Reales:7 días
Descripción: Debe calcular un conjunto reducto que permita reducir la dimensionalidad de los datos,	

facilitando el proceso de aprendizaje. Para ello necesita: Calcular el criterio de selección de rasgos MLRelevance. Calcular la heurística H(a). Calcular el grado de dependencia entre dos conjuntos de atributos. Calcular el vector de los atributos que pueden ser seleccionados como el más relevante. Calcular R(a) siguiendo el criterio MLRelevance. Determinar los n mejores atributos que maximizan R(a). Formar todas las combinaciones posibles de los *n* mejores atributos. Calcular el criterio de selección PRelevance.

Historia de Usuario	
Código:HU_#_2	**Nombre Historia de Usuario:** Construir los intervalos discretos a partir de los atributos continuos
Referencia: RF_#_10	
Programador responsable: Eden Sobrino Quiñones	Iteración: 2
Prioridad: Muy Alta	Puntos Estimados: 5 días
Riesgo en Desarrollo: Alta	Puntos Reales:5 días
Descripción: Debe convertir los atributos continuos en discretos, o sea construir los intervalos discretos para cada atributo en los que se puedan agrupar todos los valores de un atributo continuo. En este paso se sugieren tres posibles variantes: Especificación manual por un experto. Seguir un proceso sencillo de discretización de los atributos, tomando como estrategia: la construcción de intervalos de igual tamaño o de igual frecuencia. Llevar a cabo un proceso de clusterización a nivel de atributo Adoptándose la especificación manual por un experto	

2.4.2 Lista de reserva del producto

A través de la lista de reserva del producto se definen y priorizan las funcionalidades que tendrá el sistema, además se describen los requisitos no funcionales que tendrá el software. Aunque

los requisitos funcionales y no funcionales no forman parte de los artefactos que se generan en la metodología XP, se considera que una descripción de estos podría, junto a las historias de usuario facilitar el desarrollo.

Requisitos funcionales

Los requisitos funcionales expresan la esencia y definen el funcionamiento del software, establecen como el sistema debe reaccionar a una entrada en particular y como debe comportarse ante tal situación, es el conjunto de funcionalidades que va a realizar la aplicación para automatizar un proceso determinado (Sommerville, 2007). Los requisitos funcionales a implementar hacen un total de 21 y se pueden encontrar dentro de los artefactos adjuntos al documento de tesis, a continuación se listan algunos de los principales:

Código	Descripción	Prioridad
	Determinar conjunto de atributos que constituya un reducto	Muy Alta
RF_#_1	Determinar conjunto de atributos que constituya un reducto.	Muy Alta
RF_#_2	Calcular el criterio de selección de rasgos MLRelevance.	Muy Alta
RF_#_3	Calcular la heurística H(a).	Muy Alta
RF_#_4	Calcular el grado de dependencia entre dos conjuntos de atributos.	Muy Alta
RF_#_5	Calcular el vector de los atributos que pueden ser seleccionados como más relevantes.	Muy Alta
RF_#_6	Calcular R(a) siguiendo el criterio MLRelevance.	Muy Alta
RF_#_7	Determinar los n mejores atributos que maximizan R(a).	Muy Alta
RF_#_8	Formar todas las combinaciones posibles de los **N** mejores atributos.	Muy Alta
RF_#_9	Calcular el criterio de selección PRelevance.	Muy Alta
	Construir los intervalos discretos a partir de los atributos continuos	Muy Alta
RF_#_10	Construir los intervalos discretos a partir de los atributos continuos	Muy Alta
	Construir las variables lingüísticas para cada uno de los atributos continuos	Muy Alta
RF_#_11	Construir la variable lingüística para cada atributo continuo.	Muy Alta
RF_#_12	Definir los términos lingüísticos de los conjuntos difusos para cada variable lingüística.	Muy Alta
RF_#_13	Construir las funciones de pertenencia para cada conjunto difuso.	Muy Alta
	Sustituir valores numéricos por el término lingüístico mayor grado de pertenencia	Muy Alta

RF_#_14	Calcular el grado de pertenencia de un valor a un conjunto difuso.	Muy Alta
RF_#_15	Sustituir un valor numérico por el término lingüístico del conjunto difuso al que mayor grado de pertenencia presenta.	Muy Alta
	Generar reglas a partir del conjunto de entrenamiento	Muy Alta
RF_#_16	Determinar si un caso es cubierto por una regla.	Muy Alta
RF_#_17	Crear la base de reglas difusas.	Muy Alta
RF_#_18	Construir un antecedente de una regla.	Muy Alta
RF_#_19	Construir un consecuente de una regla.	Muy Alta
RF_#_20	Generar una nueva regla.	Muy Alta
RF_#_21	Marcar un caso como visitado.	Muy Alta

Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que los servicios o funciones del sistema a desarrollar deben poseer al final de su elaboración. Estas propiedades no están directamente relacionadas con las funcionalidades que se derivan del negocio. Le confieren al producto final ciertas características como usabilidad, rapidez, confiabilidad y seguridad. Por lo general son aplicados al sistema en su totalidad (Sommerville, 2007). Los requisitos no funcionales necesarios para la óptima ejecución son:

Código	Software
RNF_#_1	En el servidor de aplicación para el correcto funcionamiento es necesario tener instalado alguna distribución del Sistema Operativo GNU/Linux, el servidor web Apache2.0, la suite GESPRO, la plataforma de desarrollo R y la extensión del lenguaje procedural *plr.*
RNF_#_2	El servidor de datos, debe estar instalado el Gestor de Base de Datos PostgreSQL9.1 o superior y la herramienta para la administración de la base de datos PgAdmin III.
	Hardware
RNF_#_3	Para garantizar un buen desempeño de la aplicación en los servidores se necesita una computadora con 2 GB de memoria RAM o superior, un disco duro 320 GB o superior por el volumen de datos que se generan, un procesador Dual Core 2.0 GHz o superior.

2.4.3 Plan de iteraciones

Después de identificar las historias de usuario y estimar el esfuerzo para la realización de las mismas se prosigue a confeccionar el plan de iteración para la planificación de la etapa de implementación del sistema. Este plan define cuales historias de usuario serán implementadas en cada iteración y que tiempo se estima que durará el desarrollo de cada una de ellas. El sistema será desarrollado en las siguientes cinco iteraciones:

Iteración	Orden de la HU a Implementar	Duración HU (días)	Duración Total (semanas)
1	Determinar conjunto de atributos que constituya un reducto	9	1,2
2	Construir los intervalos discretos a partir de los atributos continuos	5	0.7
3	Construir las variables lingüísticas para cada uno de los atributos continuos	7	1
4	Sustituir valores numéricos por el término lingüístico mayor grado de pertenencia	7	1
5	Generar reglas a partir del conjunto de entrenamiento	10	1,4
Total			5,3

2.5 Fase de diseño

Durante la fase de diseño se confeccionan las tarjetas clase responsabilidad colaborador para la descripción de cada una de las entidades, se realiza el diagrama de clases para tener mejor visión de la estructura del sistema y se diseña el modelo de datos.

2.5.1 Tarjetas Clase-Responsabilidad-Colaborador (CRC)

Durante la fase de diseño se elaboran las tarjetas clase-responsabilidad-colaborador (CRC). El uso de este tipo de tarjetas es una técnica de modelado que permite identificar las clases, sus atributos y responsabilidades. El objetivo es obtener un diseño simple, elegante y fácil de comprender por parte de los programadores. A continuación se muestran dos de las seis tarjetas elaboradas, el resto se encuentra en los anexos (véase Anexo 12).

tsoftcomputing_dfuzzysets	
Descripción: Almacena los conjuntos difusos	
Atributos	
Nombre	Descripción
id	Campo identificador
id_attribute	Identificador del atributo
name_variable	Nombre del atributo
linguistic_term	Término lingüístico
membership_type	Membresía
lim_low	Límite inferior del conjunto
lim_up	Límite superior del conjunto
Responsabilidades	
Nombre	Colaborador

tsoftcomputing_nattribute	
Descripción: Almacena los atributos	
Atributos	
Nombre	Descripción
id	Campo identificador
var_id	Identificador
var_name	Nombre
tipo_dato	Tipo de dato
Responsabilidades	
Nombre	Colaborador

2.5.2 Diagrama entidad

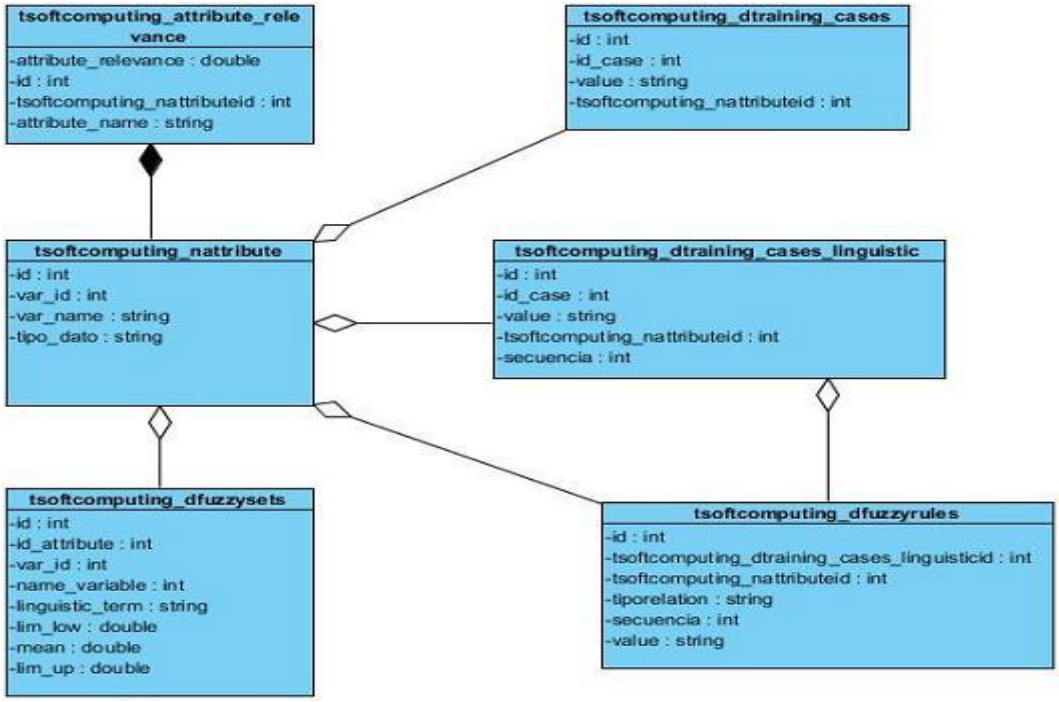


Figura 12. Diagrama de entidad a partir de las tarjetas CRC (del autor)

2.5.3 Modelo de datos

Figura 13. Modelo de datos (del autor)

2.6 Fase de codificación

En esta fase se planifican y ejecutan las tareas de programación, durante la misma se codifica todo el sistema diseñado obteniendo como resultado la herramienta.

2.6.1 Tareas de programación

Las tareas de programación son actividades sencillas que se derivan de las historias de usuario para simplificar la implementación de las mismas, se plasman en tarjetas de papel donde se describe que se debe realizar y son muy dinámicas y flexibles, pueden ser cambiadas por otras más generales o más específicas, agregarse nuevas o modificarse según las necesidades existentes. Cada una de estas tareas podrá ser comprobada a través de los casos de prueba.

A continuación se relacionan las tareas a desarrollar para la implementación del algoritmo propuesto, y la descripción de 2 de ellas, el resto puede ser consultada en Anexo 11.

Historia de Usuario	Tareas de Programación
Determinar conjunto de atributos que constituya un reducto	Determinar conjunto de atributos que constituya un reducto.
	Calcular el criterio de selección de rasgos MLRelevance.
	Calcular la heurística H(a).
	Calcular el grado de dependencia entre dos conjuntos de atributos.
	Calcular el vector de los atributos que pueden ser seleccionados como más relevantes.
	Calcular R(a) siguiendo el criterio MLRelevance.
	Determinar los n mejores atributos que maximizan R(a).
	Formar todas las combinaciones posibles de los **N** mejores atributos.
	Calcular el criterio de selección PRelevance.
Construir los intervalos discretos a partir de los atributos continuos	Construir los intervalos discretos a partir de los atributos continuos
Construir las variables lingüísticas para cada uno de los	Construir la variable lingüística para cada atributo continuo.
	Definir los términos lingüísticos de los conjuntos difusos para cada variable lingüística.

atributos continuos	Construir las funciones de pertenencia para cada conjunto difuso.
Sustituir valores numéricos por el término lingüístico mayor grado de pertenencia	Calcular el grado de pertenencia de un valor a un conjunto difuso.
	Sustituir un valor numérico por el término lingüístico del conjunto difuso al que mayor grado de pertenencia presenta.
Generar reglas a partir del conjunto de entrenamiento	Determinar si un caso es cubierto por una regla.
	Crear la base de reglas difusas.
	Construir un antecedente de una regla.
	Construir un consecuente de una regla.
	Generar una nueva regla.
	Marcar un caso como visitado.

Tarea	
Número Tarea: HU_#_1-8	Historia de Usuario: HU_#_1
Nombre Tarea: Formar todas las combinaciones posibles de los **N** mejores atributos	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1
Fecha Inicio: 4/5/2014	Fecha Fin: 4/5/2014
Programador Responsable: Eden Sobrino Quiñones	
Descripción: Formar todas las combinaciones de N en p con los atributos seleccionados como mejores. Como resultado de este paso obtenemos el conjunto formado por la unión de todas las posibles combinaciones.	

Tarea	
Número Tarea: HU_#_2-1	Historia de Usuario: HU_#_2
Nombre Tarea: Construir los intervalos discretos a partir de los atributos continuos	
Tipo de Tarea : Desarrollo	Puntos Estimados: 5
Fecha Inicio: 6/5/2014	**Fecha Fin:** 10/5/2014

Programador Responsable: Eden Sobrino Quiñones
Descripción: Debe convertir los atributos continuos en discretos, o sea construir los intervalos discretos para cada atributo en los que se puedan agrupar todos los valores de un atributo continuo. En este paso se sugieren tres posibles variantes: • Especificación manual por un experto • Construcción de intervalos de igual tamaño o de igual frecuencia • Llevar a cabo un proceso de clusterización a nivel de atributo

2.6.2 Estándares de codificación

El uso de estándares o reglas de codificación al comenzar la implementación de la herramienta, trae para el producto final muchos beneficios. Entre estos se pueden mencionar que se genera un código muy legible, muy fácil de mantener y con un alto rendimiento a la hora de ejecutarse.

Nombres de variables, parámetros de métodos y nombre de métodos:

- Todos los nombres deben comenzar con letra minúscula.
- Como convención los nombre compuestos por varias palabras deben ser separado por el guión bajo con un guión bajo por ejemplo: histograma, vector_valores y membership_type.

Líneas y comentarios de códigos:

- R no necesita punto y coma final para decir que una línea de código terminó si estas están en líneas separadas.
- Las líneas de código tendrán siempre un máximo de 70 caracteres para facilitar comprensión de esta.
- Para comentar el código o realizar algún comentario del mismo, se utiliza el símbolo de número (#) por ejemplo:

Vamos a almacenar la frase *"Hola Mundo"* en la variable hola_mundo

hola_mundo <- "Hola mundo"

En la escritura de cada elemento de una línea de código se tendrán en cuenta un espacio entre cada elemento para una mayor legibilidad a continuación dos ejemplos:

Forma Incorrecta:

vector_valores<-na.omit(vector_valores)

Forma Correcta:

vector_valores <- na.omit (vector_valores)

2.6.3 Funcionalidades de R

A continuación se listan algunas de las funcionalidades de R más empleadas en la solución.

pg.spi.exec(): en la ejecución de una consulta SQL.

hist(): en la construcción de un histograma con intervalos de igual amplitud.

as.data.frame(): en la construcción de un marco de datos.

str_c() y **paste():** en la construcción de cadenas de caracteres.

combn(X, Y): para generar todas las posibles combinaciones de Y elementos a partir de los elementos del vector X.

str_extract_all(): para encontrar segmentos de una cadena de caracteres que se hagan corresponder con una expresión regular especificada.

2.6.4 Código para el cálculo de R(a) siguiendo el criterio de selección de rasgos MLRelevance.

Con el propósito de ejemplificar el uso dado a los estándares de codificación y las principales funcionalidades de R empleadas en el desarrollo de la solución, a continuación en Figura 14 se muestra el código escrito en la implementación del primer paso de la versión del algoritmo que responde al cálculo de la relevancia de un atributo haciendo uso del criterio de selección de rasgos MLRelevance.

```
r_a <- 0
e   <- 2.7182881828459
s <- pg.spi.exec(paste("SELECT count(",atributo,")","","FROM","", tabla,";", sep= " "))
k <- pg.spi.exec(paste("SELECT count( distinct ", "",atributo," )","","FROM","", tabla,";", sep=" "))
k <- na.omit(k)
k <- as.numeric(k)

valores_k <- pg.spi.exec(paste("SELECT distinct ", "",atributo," ","","FROM","", tabla,";", sep=" "))
valores_k <- na.omit(valores_k)
valores_k <-  as.data.frame(valores_k)

for(a in 1:k){
  valor<- valores_k[a,]
  si <- pg.spi.exec(str_c("SELECT count(","", atributo,")FROM","",tabla,"WHERE ","",atributo,"=",sep=""))
  ci <- pg.spi.exec(str_c("SELECT count(distinct", clase,")FROM",tabla,"WHERE",atributo"=",sep=""))
  r_a <- r_a + (si/s)*e^(1-ci)
      }
pg.spi.exec(paste("INSERT INTO relevancia(valor, atributo)VALUES(", r_a,",","'",atributo,"'",")",sep=""))
```

Figura 14. Código para el cálculo de la relevancia de un atributo (del autor)

2.7 Conclusiones del capítulo

En este capítulo se arribó a las siguientes conclusiones:

- Se definió la propuesta de solución del problema, detallándola con la ayuda de los artefactos propuestos por la metodología programación extrema, lo cual posibilitó organizar todo el proceso de desarrollo.
- Se describieron las funcionalidades mediante historias de usuarios, de las cuales se elaboró el plan de iteraciones para definir el momento en el que serán implementadas y cuánto durará su desarrollo.
- Se realizó una descripción de la fase de diseño donde se elaboraron las tarjetas CRC, se realizó un análisis de las buenas prácticas en el desarrollo de bases de datos necesarias en la implementación del algoritmo GenRul5.
- Finalmente durante la fase de codificación se elaboraron las tareas de programación, derivadas de las historias de usuario y para estandarizar el código generado por el equipo de desarrollo en esta fase, se definieron los estándares de código a utilizar, obteniéndose finalmente la herramienta deseada acorde a los requisitos iniciales.

Capítulo 3: VERIFICACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se describe la verificación y validación de las funcionalidades desarrolladas, esto se hará mediante las pruebas de software, con el objetivo de lograr la aceptación del cliente y comprobar que se alcanzaron los objetivos enunciados inicialmente.

3.1 Fase de Pruebas

Una de las principales fortalezas de la metodología de desarrollo XP es el proceso de prueba, el cual se realiza de manera continua para asegurar durante todo el proceso de desarrollo, el éxito del producto que se está elaborando. Esto permite elaborar con mayor calidad ya que los errores son detectados en un plazo de tiempo corto y se corrigen de una manera más sencilla.

XP divide las pruebas en dos grupos, las pruebas unitarias y las pruebas de aceptación. Las pruebas unitarias son las encargadas de verificar el código y estas son diseñadas por los programadores. Mientras que las pruebas de aceptación o pruebas funcionales están destinadas a evaluar si al final de una iteración se consiguió la funcionalidad que se esperaba y que esta esté en función de los requisitos establecidos inicialmente, estas pruebas usualmente son diseñadas por el usuario o cliente final.

El objetivo fundamental que tienen las pruebas de software, es verificar los requisitos del sistema, por lo que son los propios requisitos la principal fuente de información a la hora de construir las pruebas del sistema (Gutiérrez, y otros, 2006).

3.1.1 Pruebas unitarias

Las pruebas unitarias o pruebas de caja blanca se basa en realizar pruebas al código del sistema. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas unitarias no se le pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todo estos caminos y muchos menos se podrían procesar todos. Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos (Presman, 2010).

Uno de los métodos o técnicas de prueba unitarias, es *la prueba del camino básico*. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un

conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba. Esta técnica asegura que durante la prueba se ejecute al menos una vez cada sentencia del código que se está probado (Presman, 2010). Esta será la técnica a utilizar para desarrollar los casos de pruebas.

De igual manera existen varias métricas de software para realizar pruebas unitarias, entre estas se encuentra *la complejidad ciclomática*, la cual será utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento. La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes (cualquier camino del programa que introduce por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición) del conjunto básico (es el conjunto de caminos independientes) de un programa y nos da un límite superior para el número de casos de prueba que se deben realizar para asegurar que cada sentencia de código se ejecuta al menos una vez (Presman, 2010).

Para realizar las pruebas unitarias o de caja blanca al código de la herramienta se seleccionaron varios algoritmos que son los fundamentales entre todo el conjunto de funcionalidades que dan respuesta a la implementación del algoritmo basado en GenRul5. A continuación se explica todo el procedimiento de obtener los casos de prueba y poner en ejecución estos a través de un ejemplo, utilizando la técnica prueba del camino básico junto con la métrica complejidad ciclomática. Para esto se hará uso de una notación para representación de flujo de control, denominada grafo de flujo, véase en Anexo 8.

El algoritmo seleccionado lleva como nombre generar_reglas_difusas tiene como finalidad generar reglas difusas a partir de la base de casos, para ello hace una lectura de la tabla donde se encuentran los casos con sus valores numéricos sustituidos por los términos lingüísticos de acuerdo al principio de máxima pertenencia. Se crea la base de reglas difusas R, inicialmente R = φ. Se selecciona un caso del conjunto de entrenamiento que no esté marcado como visitado y tampoco esté cubierto por las reglas generadas y se genera la nueva regla. En Figura 15 el código del algoritmo enunciado en bloques de más de una sentencia procedimental. Cada bloque de los numerados corresponde a más de una línea de código en el algoritmo implementado, las cuales han sido simplificadas a los bloques procedimentales teniendo en cuenta que pueden ser reagrupadas en un mismo nodo las sentencias que se encuentren a un mismo nivel de ejecución sin afectar el análisis del algoritmo original.

```
A__while(cont <= cant_filas - 4){
B___if(length(vector_reglas_construidas) != 0){
C______nueva_regla <- paste(A1, "^", A2, "^",A3)
    }
D___if(length(vector_reglas_construidas) == 0){
E______for(a in 1:4){
F________pg.spi.exec(paste("INSERT INTO tsoftcomputing_dfuzzyrules VALUE"))
      }
    }
G___vector_reglas_construidas <- c(vector_reglas_construidas, nueva_regla)
}
```

Figura 15. Bloques de código de la función generar_reglas_difusas (del autor).

Seguidamente aparece el grafo de flujo confeccionado a partir del código anterior, cada nodo corresponde a la numeración de las sentencias y los nodos teñidos representan los nodos predicados del gráfico. Este grafo muestra todas las posibles rutas a seguir durante la ejecución del algoritmo generar_reglas_difusas.

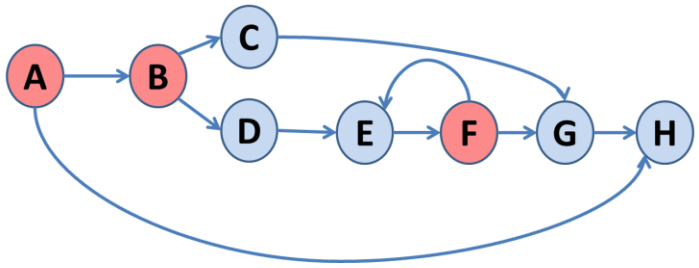


Figura 16. Grafo de flujo de la función generar_reglas_difusas (del autor).

Complejidad ciclomática del grafo.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante dos fórmulas, las cuales tienen que mostrar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática:

1. **V (G) = (A – N) + 2** Donde “A” es el número de aristas y “N” el de nodos.

2. **V (G) = P + 1** Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

Fórmula 1	**Fórmula 2**
V (G) = (A – N) + 2	V (G) = P + 1

V (G) = (8 – 6) + 2
V (G) = 4

V (G) = 3 + 1
V (G) = 4

Luego de realizado el cálculo mediante las dos fórmulas se observa el mismo valor, por lo que puede afirmarse que la complejidad ciclomática del código es 4, lo que significa que existen cuatro vías independientes por donde el flujo puede circular. Ese valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Camino 1: A-H
Camino 2: A-B-C-G-H
Camino 3: A-B-D-E-F-G-H
Camino 4: A-B-D-E-F-E-F-G-H

Después de haber obtenido los caminos básicos del flujo, se elaboran los casos de pruebas para el procedimiento por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que los parámetros obligatorios pasen nulos al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados esperados: Se expone el resultado que el procedimiento espera.

Resultados obtenidos: Se muestra el resultado luego de la ejecución del camino

La realización de los casos de prueba se ejemplificará a continuación. Mostrándose para dos de los caminos del grafo de flujo.

Caso de prueba para el Camino 1:

Descripción:

En dicho caso de prueba se verifica la estructura repetitiva *while* representada en el nodo A del grafo de flujo del código antes expuesto.

Condición de ejecución:

La estructura se estará repitiendo mientras la variable *cont* sea menor o igual a la variable *cant_filas* disminuida en cuatro unidades.

Entrada:

La variable *cont* tiene un valor de 5 mientras el valor de *cant_filas* es 8.

Resultados esperados:

Se espera que la condición cont <= cant_filas – 4 al ser comprobada no se cumpla y el algoritmo finalice su ejecución.

Resultados obtenidos:

La comprobación de la condicional no se cumplió por lo que fue satisfactoria la ejecución del caso de prueba al camino 1.

Caso de prueba para el Camino 2:

Descripción:

En dicho caso de prueba se verifica la estructura condicional *if* representada en el nodo B del grafo de flujo del código.

Condición de ejecución:

La estructura comprobará que la longitud de la variable vector_reglas_construidas sea distinta de cero.

Entrada:

La variable vector_reglas_construidas tiene un valor de 2.

Resultados esperados:

Se espera que transite por dicha condicional de modo satisfactorio.

Resultados obtenidos:

La prueba fue correcta, pues se cumplió la condicional.

3.1.2 Pruebas de aceptación

Como parte del proceso de aceptación por el cliente fueron elaborados dos conjuntos de entrenamiento con 200 y 400 casos respectivamente con datos reales de proyectos almacenados en una base de datos de proyectos terminados del laboratorio de desarrollo de GESPRO. Ambos conjuntos se encuentran compuestos por cinco indicadores relacionados con las áreas de conocimiento de la gestión de proyectos más importantes identificadas en la literatura (Lugo García, y otros, 2012):

- Índice de Rendimiento de la Ejecución (IRE).
- Índice de Rendimiento de la Planificación (IRP).
- Índice de Calidad del Dato (ICD).
- Índice de Rendimiento de los Recursos Humanos (IRRH).
- Índice de Rendimiento de la Eficacia (IREF).

Cada caso está estructuralmente formado por un antecedente comprendido por los cinco indicadores que pueden estar jerarquizados en: Alto, Medio o Bajo. Así como un consecuente que clasifica el caso en: Bien, Regular o Mal.

Conjunto 1 = 200 casos **Conjunto 2** = 400 casos

Luego de aplicado el algoritmo en ambos conjuntos de entrenamiento se obtuvieron los siguientes resultados:

Como resultado del primer paso los tres atributos más relevantes y que constituyen el reducto son:

Conjunto 1

	id integer	attribute_relevance double precision	attribute_name character varying(255)	tsoftcomputing_nattributeid integer
1	477	3.81718232715919	iref	4
2	480	2.77395348622749	irrh	2
3	478	2.05699295924394	icd	1

Conjunto 2

	id integer	attribute_relevance double precision	attribute_name character varying(255)	tsoftcomputing_nattributeid integer
1	466	2.56576086246568	icd	1
2	465	2.53730208366754	irrh	2
3	463	2.52146137277458	iref	4

Luego de sustituido los valores numéricos por los términos lingüísticos de mayor pertenencia:

Conjunto 1 **Conjunto 2**

Aparecen 2 nuevos casos Aparecen 3 nuevos casos

	count bigint
1	202

	count bigint
1	403

Lo anterior sucede porque se dio la peculiaridad de que un atributo tenía el máximo valor de pertenencia en más de un conjunto difuso de ahí que quedan generados nuevos casos donde el atributo tome cada término lingüístico de los conjuntos difusos donde es máxima su pertenencia.

Ejemplificación de los nuevos casos del Conjunto 1

Primer Caso: En el conjunto 1 el caso número 1 el atributo ICD de identificador 1 alcanza la máxima pertenencia en los conjuntos difusos cuyos términos lingüísticos se hacen corresponder con Bajo y Medio. Lo que implica la generación de dos nuevos casos.

	id integer	id_case integer	value character v	tsoftcomputing_nattributeid integer	secuencia integer
1	20494	1	Bajo	1	1
2	20495	1	Medio	2	1
3	20496	1	Medio	4	1
4	20497	1	M	8	1
5	20498	1	Medio	1	2
6	20499	1	Medio	2	2
7	20500	1	Medio	4	2
8	20501	1	M	8	2

Segundo Caso: En el conjunto 1 el caso número 86 el atributo IREF de identificador 4 alcanza la máxima pertenencia en los conjuntos difusos cuyos términos lingüísticos se hacen corresponder con Bajo y Medio. Lo que implica la generación de dos nuevos casos.

	id integer	id_case integer	value character	tsoftcomputing_nattributeid integer	secuencia integer
1	20838	86	Alto	1	87
2	20839	86	Bajo	2	87
3	20840	86	Bajo	4	87
4	20841	86	M	8	87
5	20842	86	Alto	1	88
6	20843	86	Bajo	2	88
7	20844	86	Medio	4	88
8	20845	86	M	8	88

Ejemplificación de los nuevos casos del Conjunto 2

El primer y segundo de este conjunto caso se hacen corresponder con los del Conjunto 1 respectivamente.

Tercer Caso: En el conjunto 2 el caso número 203 el atributo IREF de identificador 4 alcanza la máxima pertenencia en los conjuntos difusos cuyos términos lingüísticos se hacen corresponder con Bajo y Medio. Lo que implica la generación de dos nuevos casos.

	id integer	id_case integer	value character varying	tsoftcomputing_nattributeid integer	secuencia integer
1	19698	203	Alto	1	205
2	19699	203	Bajo	2	205
3	19700	203	Bajo	4	205
4	19701	203	M	8	205
5	19702	203	Alto	1	206
6	19703	203	Bajo	2	206
7	19704	203	Medio	4	206
8	19705	203	M	8	206

El total de reglas difusas generadas como resultado del último paso del algoritmo es para cada conjunto la que se muestra a continuación:

Conjunto 1 **Conjunto 2**

1	{{21}}

1	{{24}}

En ambos casos se obtienen menos reglas que las 27 usadas actualmente en el SIB Sugeno grado cero de GESPRO definidas según criterio de expertos.

Mientras que las reglas difusas generadas son:

Conjunto 1

	ICD	IRRH	IREF	Resultado
1	Bajo	and Medio	and Medio	→ M
2	Medio	and Medio	and Medio	→ M
3	Alto	and Bajo	and Bajo	→ B
4	Alto	and Medio	and Medio	→ M
5	Alto	and Bajo	and Alto	→ R
6	Alto	and Bajo	and Alto	→ M
7	Alto	and Bajo	and Bajo	→ M
8	Alto	and Alto	and Bajo	→ M
9	Bajo	and Bajo	and Bajo	→ M
10	Alto	and Bajo	and Medio	→ M
11	Alto	and Bajo	and Bajo	→ R
12	Alto	and Medio	and Bajo	→ B
13	Alto	and Medio	and Bajo	→ M
14	Alto	and Medio	and Medio	→ B
15	Alto	and Bajo	and Medio	→ B
16	Alto	and Bajo	and Alto	→ B
17	Alto	and Bajo	and Medio	→ R
18	Alto	and Alto	and Bajo	→ B
19	Alto	and Alto	and Medio	→ B
20	Bajo	and Bajo	and Bajo	→ B
21	Alto	and Medio	and Medio	→ R

Conjunto 2

	ICD	IRRH	IREF	Resultado
1	Bajo	and Medio	and Medio	→ M
2	Medio	and Medio	and Medio	→ M
3	Alto	and Bajo	and Bajo	→ B
4	Alto	and Medio	and Medio	→ M
5	Alto	and Bajo	and Alto	→ R
6	Alto	and Bajo	and Alto	→ M
7	Alto	and Bajo	and Bajo	→ M
8	Alto	and Alto	and Bajo	→ M
9	Bajo	and Bajo	and Bajo	→ M
10	Alto	and Bajo	and Medio	→ M
11	Alto	and Bajo	and Bajo	→ R
12	Alto	and Medio	and Bajo	→ B
13	Alto	and Medio	and Bajo	→ M
14	Alto	and Medio	and Medio	→ B
15	Alto	and Bajo	and Medio	→ B
16	Alto	and Bajo	and Alto	→ B
17	Alto	and Bajo	and Medio	→ R
18	Alto	and Alto	and Bajo	→ B
19	Alto	and Alto	and Medio	→ B
20	Bajo	and Bajo	and Bajo	→ B
21	Alto	and Medio	and Medio	→ R
22	Bajo	and Bajo	and Bajo	→ R
23	Bajo	and Bajo	and Alto	→ M
24	Alto	and Alto	and Medio	→ M

Finalmente el tiempo de ejecución del algoritmo para aprender de cada conjunto de entrenamiento expresado en milisegundos (ms) es el que se muestra a continuación:

Conjunto 1

OK.	Unix	Ln 1, Col 1, Ch 1		1 row.	21035 ms

Conjunto 2

OK.	Unix	Ln 1, Col 1, Ch 1		1 row.	80764 ms

3.2 Conclusiones del capítulo

Del presente capítulo se concluye que:

- Se realizó la verificación y validación del algoritmo desarrollado asegurando la adecuada correspondencia entre las funcionalidades desarrolladas con los requisitos iniciales.
- Se ejecutaron las pruebas unitarias y de aceptación que son las que propone la metodología XP, para comprobar en cada una de las iteraciones del desarrollo, que se han alcanzado los objetivos propuestos al inicio de estas.
- Las pruebas unitarias realizadas a varios algoritmos no detectaron errores, arrojando resultados satisfactorios.
- Finalmente se realizaron pruebas de aceptación o funcionales las cuales mediante dos casos de estudio introducidos, posibilitaron la comparación positiva de los resultados en ambos casos.

Conclusiones Generales

Del contenido presentado en el trabajo, su análisis y de los antecedentes revisados en la literatura, se concluye lo siguiente:

- Quedaron determinadas las principales tendencias en el campo del aprendizaje automatizado y su aplicación a la gestión de proyectos.
- No existen herramientas para la gestión de proyectos que hagan uso de técnicas de softcomputing para aprender las reglas que utiliza.
- Se adopta como estrategia de generación de reglas difusas la que propone la creación de métodos híbridos y dentro de ella se selecciona como algoritmo a implementar una versión del GenRul5.
- Se definió la propuesta de solución del problema, detallándola con la ayuda de los artefactos propuestos por la metodología programación extrema, lo cual posibilitó organizar todo el proceso de desarrollo.
- Se implementaron en el lenguaje procedural plr las funcionalidades de una versión del algoritmo GenRul5 que permite aprender reglas difusas de forma automatizada a partir de una base de casos.
- Fueron validados los resultados obtenidos a través de la aplicación a una base de casos de proyectos terminados en la Universidad de Ciencias Informáticas.

Bibliografía

Basogain Olabe ,Xabier .2009.*REDES NEURONALES ARTIFICIALES Y SUS APLICACIONE S.* Bilbao : Escuela Superior de Ingeniería de Bilbao, EHU , 2009.

Beck , Kent y Andres, Cynthia . 2004.*Extreme Programming Explained: Embrace Change.* 2nd Edition. 2004. 978-0321278654.

Beck, Kent. 1999.*Extreme Programming Explained.* 1999.

Bello, R y Verdegay, J L. 2010.*Los conjuntos aproximados en el contexto de la Soft Computing.* [ed.] RCCI. La Habana : s.n., 2010. págs. 5-24. Vol. 4.

Berenji, H. R. 1992.*A Reinforcement Learning-Based Architecture for Fuzzy Logic Control.* 1992. pp. 267-292.

Blanco Encinosa, Lázaro J. 2011.*La informática en la dirección de empresas.* La Habana : Editorial Félix Varela, 2011. 978-959-07-1629-4.

Botía-Blaya, J A. 2003.*Sistemas Difusos.* Murcia : Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, 2003.

Caballero, Y. 2004.*X Convención Internacional y Feria Informática .* Ciudad de la Habana : CITMATEL, 2004.

Castellano, G., et al. 2005.*Knowledge discovery by a neuro-fuzzy modeling framework.* Bari : s.n., 2005. pp. 187-207. www.elsevier.com/locate/fss.

Castro Díaz-Balart, Fidel y Delgado Fernández, Mercedes. 1999.*INNOVACION TECNOLOGICA, ESTRATEGIA CORPORATIVA Y COMPETITIVIDAD EN LA INDUSTRIA CUBANA.* Madrid : Universidad Politécnica de Madrid, 1999. 2171-6323.

Chen, G., Liu, D. and Li, J. 2001.*Influence and conditional influence—new interestingness measures in association rule mining.* Vancouver : s.n., 2001.

Cho, Jae-Soo and Park, Dong-Jo. 2000.*Novel fuzzy logic control based on weighting of partially inconsistent rules using neural network.* 2000. pp. 99–110. ISSN 1064-1246.

Cox, Earl, Taber, RodMan y O'Hagan, Michael, O'Hagen, Michael. 1998.*The Fuzzy Systems Handbook, Second Edition.* 2nd Bk&Cd edition. New York : AP Professional, Paperback, 1998. pág. 589.

Dataprix. 2013. Dataprix. [En línea] 2013. [Citado el: 4 de diciembre de 2013.] http://www.dataprix.com/8-cliente-grafico-pgadmin3.

Delgado Victore, Roberto, et al. 2011.*La Dirección Integrada de Proyecto como Centro del Sistema de Control de Gestión en el Ministerio del Poder Popular.* Caracas : CENDA, 2011.

Fabri, Jose Augusto. 2002.*A Neuro-Fuzzy system to discover knowledge (NeuroFAC).* Fakultat fur Informatik, Otto-von-Guericke. Magdeburg : s.n., 2002. p. 97. Tesis de Doctorado.

Fabri, Jose Augusto and Camargo, Heloisa de Arruda. 2000.*Sistema Neuro-Fuzzy para Aquisição de Conhecimento (NeuroFac).* Spain : s.n., 2000. pp. 284-293.

Fernández Ramírez, José Luis. 2012. Lider de Proyecto. com. [En línea] 2012. [Citado el: 27 de mayo de 2013.] http://www.liderdeproyecto.com/articulos/introduciendo_a_prince2.html.

George, R. y Srikanth, R. 1996.*Data summarization using genetic algorithms and fuzzy logic.* s.l. : Physica-Verlag,Heidelberg, 1996. págs. 599–611.

Golberg, D. 1989.*Genetic Algorithms in Search, Optimization, and Machine Learning.* s.l. : Addison-Wesley, 1989.

Grzymala-Busse, Jerzy. 1992. Lers-A system for learning from examples based on Rough Sets. *Intelligence Decision Support. Handbook of applications and advances of the RST.* s.l. : Kluwer Academics, 1992.

Gutiérrez, JJ, y otros. 2006.*Pruebas del Sistema en Programación Extrema.* Sevilla. 2006 : Departamento de Lenguajes y Sistemas Informáticos.Universidad de Sevilla, 2006.

Gutiérrez-Martínez, I. 2003.*Un Modelo para la Toma de Decisiones usando Razonamiento Basado en Casos en condiciones de Incertidumbre.* Santa Clara : Departamento de Ciencias de la Computación, Facultad de Matemática-Física y Computación. Universidad Central Marta Abreu de las Villas, 2003. pág. 116.

Halgamuge, S. K. and Glesner, M. 1994.*Neural Networks in Designing Fuzzy Systems for real World Applications.* 1994. pp. 1-12.

Hamilton, Kim y Russell, Miles. 2006.*Learning UML 2.0.* s.l. : O'Reilly, 2006, 2006. 0-596-00982-8.

Herrera, Francisco. 2001.*Sistemas Neuro-difusos.* Acapulco : Dpto. Automática y Sistemas Computacionales, Universidad Central "Marta Abreu" de las Villas, 2001. pág. 69.

Holland, J. H. and Reitman, J. S. 1978. Cognitive systems based on adaptive algorithms. [ed.] D. A. Waterman and F. Hayes-Roth. *Pattern-Directed Inference Systems.* New York : Academic Press, 1978, pp. 313–329.

Hong, Tzung-Pei y Lee, Chai-Ying. 1996.*Induction of fuzzy rules and membership functions from training examples.* 1996. págs. 33-47.

Hong, Tzung-Pei, Lin, Kuei-Ying and Wang, Shyue-Liang. 2003.*Fuzzy data mining for interesting generalized association rules.* 2003. pp. 255-269.

Humphrey, W. S. 2000.*The Team Software Process (TSP).* Pittsburgh: Carnegie Mellon : Software Engineering Institute, 2000.

IPMA. 2006.*ICB - IPMA Competence Baseline, Version 3.0.* s.l. : International Project Management Association, 2006. 0-9553213-0-1.

ISO. 2012.*ISO 21500 Guidance on project management.* 2012.

Ivar, J, Grady, B y James, R. 2000.*El Proceso Unificado de Desarrollo e Software.* Primera . s.l. : Pearson Educación S.A, 2000.

Jang, J. S. 1993.*ANFIS: Adaptative-Network-Based Fuzzy Inference Systems.* 1993. pp. 665-685.

Jang, J. S., Sun, C. and Mizutani, E. 1997.*Neuro fuzzy and Soft Computing.* New York : Prentice Hall, 1997.

Janikow, C.Z. y Faifer, M. 1999.*Fuzzy Partitioning with FID3.1.* s.l. : IEEE, 1999. págs. 467–471.

KABACOFF, ROBERT I. 2012.*R in Action. Data analysis and graphics with R.* New York : Manning Publications Co, 2012, 2012.

KACPRZYK, J y ZADROŻNY, S. 2010.*Towards human consistent data driven decision support systems using verbalization of data mining results via linguistic data summaries.* Warsaw, Poland : POLISH ACADEMY OF SCIENCES, 2010. DOI: 10.2478/v10175-010-0034-2.

Kacprzyk, J., Yager, R.R. and Zadrożny, S. 2000.*A fuzzy logic based approach to linguistic summaries of databases.* 2000. pp. 813-834.

Kacprzyk, Janusz and Yager, Ronald R. 2001.*LINGUISTIC SUMMARIES OF DATA USING FUZZY.* 2001. pp. 133 — 154.

Kacprzyk, Janusz and Zadrozny, Sławomir. 2005.*Linguistic database summaries and their protoforms: towards natural language based knowledge discov ery tools.* s.l. : Elsevier Inc., 2005. pp. 281–304.

Kaya, Mehmet, y otros. 2002.*Efficient Automated Mining of Fuzzy Association Rules.* [ed.] R. Cicchetti. s.l. : Springer-Verlag Berlin Heidelberg, 2002. págs. 133-142. Vol. 2453.

KDnuggets. 2012.*[En línea] http://www.kdnuggets.com/2012/05/top-analytics-data-mining-big-data-software.html.* 2012.

Klose, A A. 2003.*Partially Supervised Learning of Fuzzy Classification Rules, in Facultat fur Informatik.* Magdeburg : Otto-von-Guericke, 2003. pág. 180.

Kohonen, T. 1988.*Self-Organization and Associative Memory.* New York : Springer-Verlag, 1988.

Komorowski, Jan, y otros. 1999. A Rough Set Perspective on Data and Knowledge. [aut. libro] J, Zytkow W. Klosgen. *The Handbook of Data Mining and Knowledge Discovery.* Oxford : Oxford University Press, 1999.

Labriet, Thierry. 2013.*Comparing PMBOK Guide 4th Edition, PMBOK Guide 5th Edition and ISO 21500.* Lausanne, Switzerland : STS Sauter Training & Simulation SA, 2013.

Larman, Craig. 1999.*UML y Patrones Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999.

Larrañaga, P y Lozano, J.A. 2002.*Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation.* Boston : Kluwer Academic, 2002. 375.

Li, Rui-Ping, Mukaidono, Masao and Turksen, I. Burhan. 2002.*A Fuzzy Neural Network for Pattern Classification and Feature Selection.* 2002. pp. 20-35.

López de Mántaras, R. 1991.*A Distance-Based Attribute Selection Measure for Decision Tree Induction. Machine Learning.* 1991.

Lugo García, Jose Alejandro y García Perez, Ana Maria. 2011.*Colección automática de métricas hacia un repositorio de mediciones.* Antioquia : s.n., 2011. págs. 199-207.

Lugo García, José Alejandro, Piñero Pérez, Pedro Y. y Delgado Victore, Roberto. 2012.*Modelo para el control de la ejecución de proyectos basado en indicadores y lógica borrosa.* La Habana : Universidad de las Ciencias Informticas, 2012.

Lugo García, José Alejandro, y otros. 2013.*Control de la ejecución de proyectos basado en indicadores y lógica borrosa.* 2013. págs. 15-35. ISSNe 2346-9161.

Maguiña Pérez, R. A. 2010.*Sistemas de inferencia basados en Lógica Borrosa: Fundamentos y caso de estudio.* 2010. págs. 91-104.

Mamdani, E. H. 1977.*Applications of fuzzy logic to approximate reasoning using linguistic synthesis.* s.l. : IEEE Transactions on Computers, 1977. págs. 1182–1191.

Man, K, Kwong, S y Tang, K. 1999.*Genetic Algorithms.* London : Springer-Verlag, 1999.

Martínez, Mariluz, y otros. 2003.*Mining Association Rules Using Fuzzy Inference on Web Data.* [ed.] E. Menasalvas. s.l. : Springer-Verlag Berlin Heidelberget, 2003. págs. 73–82. Vol. 2663.

Mitchell, Tom. 1997.*Machine Learning.* New York : McGraw-Hill Science/Engineering/Math, 1997. ISBN: 0070428077.

Moreno, Antonio. 1994.*Aprendizaje Automatico.* Barcelona : UPC, Universitat Politecnica de Catalunya. 321., 1994.

Nauck, D. and Kruse, R. 1998.*How the Learning of Rule Weights Afects the Interpretability of Fuzzy Systems.* Anchorage : s.n., 1998. pp. 1235-1240.

—. 1999.*Neuro-Fuzzy Sytems for Function Aproximation.* 1999. págs. 261-271.

Nauck, Detlef. 2000.*Data Analysis with Neuro - Fuzzy Methods.* Fakultat fur Informatik, Otto-von-Guericke. Magdeburg : s.n., 2000. p. 154. Tesis de Doctorado.

Nauck, Detlef, Klawonn, Frank y Kruse, Rudolf. 1999.*Foundations of Neuro-Fuzzy systems.* New York : Wiley, 1999.

Nomura, H., Hayashi, I. and Wakami, N. 1992.*A Learning Method of Fuzzy Inference Rules by Descent Method.* San Diego, CA. : s.n., 1992. pp. 203-210.

OGC. 2009.*Managing Successful Projects with PRINCE2.* s.l. : TSO, 2009. 978011310593.

Pal, S.K. 1997.*A new version of the rule induction LERS.* 1997. págs. 27-39.

Pal, Sankar K., Mitra, Sushmita y Mitra, Pabitra. 2001.*Rough Fuzzy MLP: Modular Evolution, Rule Generation and Evaluation.* 2001. págs. 56-69.

Palacio, Juan. 2007.*Flexibilidad con SCRUM.* 2007.

Paradigm, Visual. 2013. Visual Paradigm Characteristics. *Visual Paradigm.* [En línea] 2013. [Citado el: 8 de diciembre de 2013.] http://www.visual-paradigm.com..

Pawlak, Z. 1991.*Rough Sets- Theorical Aspects of Reasoning about Data.* 1991.

PCC. 2011.*Lineamientos de la Política Económica y Social del Partido y la Revolución.* La Habana : s.n., 2011.

Pérez Pupo, Iliana, y otros. 2013.*MODELO PARA EL APRENDIZAJE AUTOMÁTICO APLICACIÓN EN LA DIRECCIÓN INTEGRADA DE PROYECTOS.* La Habana : Ediciones pensando el futuro, 2013. pág. 10. ISBN: 978-959-7213-02-4.

Piñero Pérez, P.Y. 2013.*GESPRO Paquete para la gestión de proyectos.* La Habana : GECYT, 2013. págs. 45-53. 1682-2455.

Piñero Pérez, Pedro y et.al. 2010.*Paquete de Herramientas para la Gestión de Proyectos GESPRO. 1540-2010* Cuba, 2010. Software.

Piñero Pérez, Pedro Yobanis. 2005.*Modelo para el aprendizaje y la clasificación automática basado en técnicas de soft computing.* Santa Clara : Universidad Martha Abreu, 2005.

Piñero, P. 2000.*Biblioteca de componentes COM , DCOM para el trabajo con Algoritmos Genéticos.* Santa Clara : Departamento de Ciencias de la Computación. 2000, Universidad Central "Marta Abreu" de Las Villas, 2000. pág. 107.

—. 2001.*GACOM Biblioteca de Componentes de Algoritmos Genéticos.* Matanzas, Cuba : Universidad de Matanzas, COMAT 2001, 2001.

Piñero, Pedro y Sanfeliu, A. 2003.*Recognition, Two New Metrics for Feature Selecction in Pattern.* Berlin : Springer-Verlag, 2003. págs. 488-497. Vol. 2905.

Piñero, Pedro, y otros. 2008. Propuesta de modelo pedagógico para Maestría de Gestión de Proyectos Informáticos. *Revista Cubana de Ciencias Informáticas.* La Habana : s.n., 2008. Vol. 2, 3-4. ISSN: 1994-1536.

PMI. 2009.*Guía de los Fundamentos para la Gestión de Proyectos.* Newtown Square, Pennsylvania : Project Management Institute, Inc., 2009. 978-1-933890-72-2.

—. 2013.*Guía de los Fundamentos para la Gestión de Proyectos 5ta ed.* 5ta ed. Newtown Square, Pennsylvania : Project Management Institute, 2013. 9781935589679.

PostgreSQL. 2010.*PostgreSQL. PostgreSQL. [En línea] 2 de octubre de 2010. [Citado el: 4 de diciembre de 2013.] http://www.postgresql.org.es/sobre_postgresql#caracteristicas.* 2010.

Presman, R. 2010.*Ingenieria de Software un enfoque práctico.* Quinta . 2010. pág. 601.

Quinlan, J. R. 1996.*Improved Use of Continuous Attributes in C4.5.* 1996. pp. 77-90.

Quinlan, J. R. 2000. C5.0: An Informal Tutorial. [En línea] 2000. www.rulequest.com.

—. 1986.*Induction of Decision Trees.* 1986. págs. 81 - 106.

—. 2002.*See5/C5.0.* www.rulequest.com. : s.n., 2002.

RAE. 2014.*Real Academia de la Lengua Española.* 2014.

Ruiz Shulcloper, J, Alba-Cabrera, E y Sánc, G. 2000.*Discovering b 0 -density connected components from large mixed incomplete data sets.* Ciudad de la Habana : GEOINFO'2000, 2000.

Sánchez, Luciano and Otero, José. 2004.*A fast genetic method for inducting descriptive fuzzy models.* 2004. pp. 33-46.

SEI. 2010.*CMMI for Dev v1.3.* Pittsburgh : Carnegie Mellon University, 2010.

Selfridge, O.G. 2004.*Machine Learning.* California, USA : American Association for Artificial Intelligence, 2004.

Setiono, Rudy. 1998. Techniques for Extracting Rules from Artificial Neural Networks. [En línea] 1998. http://www.comp.nus.edu.sg/~rudys/publications.html.

Setiono, Rudy y Kheng, Wee. 2000.*An Algorithm for Fast Extraction of Fuzzy Rules from Neural Networks.* 2000. págs. 15-25.

Shi-Quan, C, Guota , M.M y Yamakaw, T. 1988.*Fuzzy Equivalence and Multiobjective Desicion Making, in Fuzzy Computing.* North Holland : Elsevier Science Publisher, 1988.

Smith, S. F. 1980.*A Learning System Based on Genetic Adaptive Algorithms.* Department of Computer Science, University of Pittsburgh. Pittsburgh : s.n., 1980. Tesis de Doctorado.

Sommerville, I. 2007.*Software Engineering.* [ed.] Harlow. Octava . Sommerville, I. 2007. Software Engineering. [ed.] Addison- Wesley. Octava Edición. s.l. : Harlow: Pearson Education. SA, 2007. : Addison- Wesley Pearson Education, 2007.

Stang, D B. 2010.*IT Project & Portfolio Management Magic Quadrant.* Stamford USA : Gartner Inc., 2010.

Suárez, Armando. 2008. Aprendizaje Automático. *Repositorio Institucional de la Universidad de Alicante.* [En línea] 2008. http://hdl.handle.net/10045/3878.

Takagi, T y Sugeno, M. 1985.*Fuzzy identification of systems and its application to modeling and control Transactions on Systems, Man, and Cybernetics.* s.l. : IEEE, 1985. págs. 116–132.

Torres López, Surayne, y otros. 2013. Experiencias en la formación de masters en gestión de proyectos de tecnologías de la información. 2013. Vol. 4, 1, págs. 63-79. ISSNe 2346 9161.

Venturini, G. 1993.*SIA: A supervised inductive algorithm with genetic search for learning attribute based concepts.* Vienna : s.n., 1993. pp. 280–296.

Wang, L..X. y Mendel, J.M. 1992.*Generating fuzzy rules by learning from examples.* 1992. págs. 1414-1427.

Wanga, Shi-tong, Yub, Dong-jun y Yangb, Jing-yu. 2003.*Integrating rough set theory and fuzzy neural network to discover fuzzy rules.* 2003. págs. 59–73.

Xizhao, Wang y Jiarong, Hong. 1998.*On the handling of fuzziness for continuous valued attributes in decision tree generation.* 1998. págs. 283-290.

Yager, R.R. 1982.*A new approach to the summarization of data.* 1982. págs. 69–86.

—. 1988.*On ordered weighted averaging operators in multicriteria decision making.* 1988. págs. 183–190.

Zadeh, L. A., I.S. 9, Editor. 1976.*The concept of a linguistic variable and its applications to approximate reasoning.* 1976. págs. 43-80.

Zadeh, L.A. 1983.*A computational approach to fuzzy quantifiers in natural languages.* 1983. pp. 149-184.

—. 2002.*A prototype-centered approach to adding deduction capabilities to search engines-the concept of a protoform.* Berkeley : University of California, 2002.

—. 1965.*Fuzzy Sets.* 1965. págs. 338-353.

—. 1973.*Outline of A New Approach to the Analysis of Complex Systems and Decision.* 1973. págs. 28-44.

Zandhuis, Anton y Stellingwerf, Rommert. 2013.*ISO21500: Guidance on project management - A Pocket Guide.* Zaltbommel : Van Haren Publishing, 2013. 978 90 8753 010 5.

Zhou, Zhi-Hua, Jiang, Yuan y Chen, Shi-Fu. 2003.*Extracting Symbolic Rules From Trained Neural Network Ensembles.* 2003. 0921-7126.