

Estrategia de Evaluación de la Calidad de Software para la etapa de mantenimiento

Nelly Condori-Fernández

Instituto de Informática de la Universidad
Nacional de San Agustín

Arequipa- Perú

nelly@unsa.edu.pe

Ernesto Cuadros

Universidade de São Paulo Av. Trabalhador
São-Carlense, 400 – Centro - Cx. Postal 668

Saint Carlos – Brazil

ecuadros@icmc.usp.br

Resumen

El presente artículo tiene como fin proponer una estrategia de evaluación de la calidad interna del software, el cual sigue los lineamientos principales de la norma ISO/IEC:9126, con la adecuación de métricas. Un sistema de inferencia neurodifuso es propuesto con el objetivo de realizar el proceso de agregación dentro de un proceso de evaluación general. Dicho sistema incorpora los conceptos de aprendizaje de las redes neuronales a los sistemas de inferencia difuso. El modelo de red neuronal conocido como FAM (Fuzzy Associative Memory) es utilizado para el almacenamiento y evaluación de las reglas difusas. De esta manera se obtiene una evaluación con un mayor grado de objetividad y flexibilidad, logrando incrementar el nivel de confiabilidad en el proceso de evaluación durante la etapa de mantenimiento.

1. Introducción

Durante mucho tiempo, la funcionalidad ha sido la única manera de medir la calidad del software. Diferentes modelos de calidad han sido propuestos y han sido presentados para su uso [11][1]. Pero, era necesario la llegada de un modelo estándar que estableciera las características fundamentales de calidad del software y que ganara el consenso para su aplicación.

Es por esta razón que el comité técnico de la ISO/IEC comenzó un trabajo conjunto para lograr el consenso requerido y alentar la normalización del Aseguramiento de Calidad del Software sobre una base internacional. Como resultado de dicho trabajo, en diciembre de 1991, se publicó la norma internacional ISO/IEC 9126:1991, [9] la cual establece los lineamientos generales para la evaluación del producto de software a partir de seis características de calidad.

El alcance de esta norma resulta limitado, ya que no contiene lineamientos sobre métodos de medición. Resulta necesario identificar un conjunto apropiado de métricas para medir atributos de calidad de software. No obstante, el uso de dichas métricas no elimina la necesidad del juicio humano en el proceso de evaluación. Esto se afirma en la siguiente sentencia: *“El uso de métricas de software reduce la subjetividad en la evaluación de la calidad de la calidad del software”* [8]. La evaluación es fundamental para cualquier área de la ingeniería y la ingeniería de software no debe ser una excepción. Actualmente aún no existe un conjunto de estrategias o métodos bien establecidos y ampliamente aceptados a nivel internacional para evaluar la calidad del producto del software. Por tal razón, hoy en día el control sobre dicha calidad es materia de preocupación no sólo para la comunidad científica de la ingeniería del software, si no también para investigadores de otras áreas.

Se han venido desarrollando trabajos con la intención de evaluar la calidad del software, aplicando técnicas de inteligencia artificial con la finalidad de representar la subjetividad que presenta el ser humano a la hora de tomar una decisión durante un proceso de evaluación. Dentro de estos trabajos podemos destacar esfuerzos presentados por [5],[2],[3]. Cabe mencionar que en el trabajo [3], se destacó una desventaja de los sistemas de inferencia difuso, pues a medida que se iba incrementando el número de entradas al sistema (atributos de calidad), el número de reglas difusas se incrementaba dramáticamente, lo que ocasionaba dificultades en el proceso de inferencia. Esta desventaja que presentan los sistemas de inferencia difuso se ven superados con un sistema de inferencia neurodifuso que combinan las ventajas de las redes neuronales y de la lógica difusa, logrando obtener un sistema más robusto, confiable y flexible de apoyo al auditor de software.

El trabajo se inicia delimitando el problema, puesto que la evaluación de la calidad del software involucra diversos aspectos a considerar. En la sección 3, se detalla el principio y estructura del sistema de inferencia neurodifuso, donde se explicará la arquitectura del sistema propuesto. Finalmente en la sección 4, se muestran conclusiones de este trabajo.

2. Delimitación del problema

Existen fundamentalmente dos enfoques que pueden ser seguidos para asegurar la calidad del software. El primer enfoque es asegurando el proceso de desarrollo del software mediante la utilización de métodos, técnicas y herramientas de la ingeniería del software que permita incorporar los atributos de calidad durante todas las etapas de desarrollo del software. El segundo enfoque es realizando la evaluación de la calidad del producto final de cada etapa. La normativa ISO 9126 se encarga de establecer características de calidad para evaluar el software como producto. Dichas características son: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. [9]. Así mismo la norma ISO 14764 determina que el PAS (Proceso de Auditoría del Software) da soporte al PMS (Proceso de Mantenimiento del Software). Se entiende por mantenimiento del software como un conjunto de actividades de la ingeniería del software que se realizan una vez que el producto ha sido entregado al cliente y puesto en operación [14].

Por lo tanto, el presente trabajo está basado en este segundo enfoque; proponer una estrategia de evaluación para la etapa de mantenimiento dentro de un PAS.

3. Arquitectura del sistema

El proceso de evaluación, comprende básicamente de tres módulos principales:

- 1.- Especificación de los criterios de evaluación, según la norma ISO/IEC 9126-1.
- 2.- Proceso de Cuantificación, mediante la adecuación de las métricas definidas en la norma ISO 9126-3.
- 3.- Proceso de Agregación, basado en un sistema de inferencia neurodifuso.

De esta manera se puede cuantificar la calidad de un producto de software, por la apropiada agregación y cuantificación de todas los atributos, elementos medibles, a partir de una medición directa o indirecta y la posterior agregación tanto a nivel de subcaracterísticas como características. (Vea figura 1)

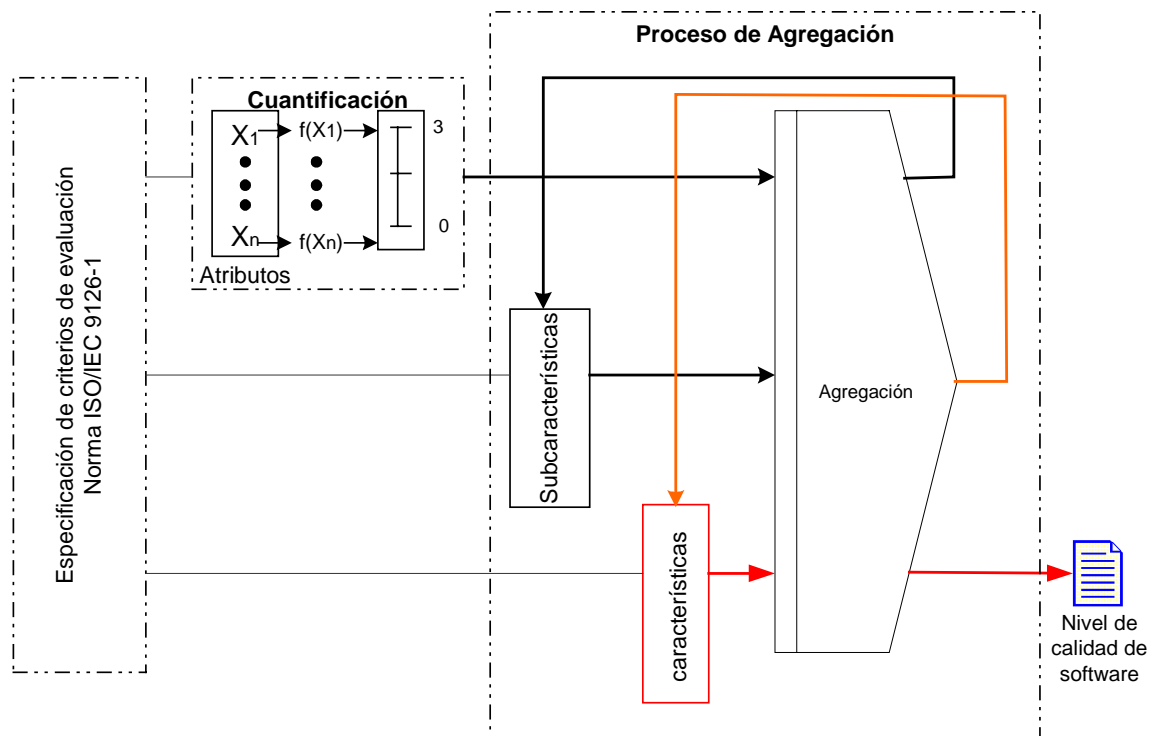


Fig. 1. Esquema general propuesto para el proceso de evaluación de la calidad del software.

A continuación se explicará el proceso de cuantificación.

3.1 Proceso de Cuantificación

Como se dijo, la calidad del software no puede ser medida como una cantidad escalar única, pero sí que está compuesta por un conjunto de características que describen los requerimientos de calidad del software. En la tabla 1, se presenta dicho conjunto de características correspondientes a la *etapa de mantenimiento del software* (funcionalidad, confiabilidad y mantenibilidad)[6]. Cada característica es refinada en múltiples subcaracterísticas de calidad que demuestren su capacidad para satisfacer los requerimientos implícitos y explícitos preestablecidos por el desarrollador y por el usuario respectivamente. Los elementos de evaluación de más bajo nivel de refinamiento son

los atributos de calidad que son las unidades más detalladas. En este último nivel, es que se aplica una serie de métricas adecuadas a un tipo de escala de valores en un intervalo de 0 a 3. En la presente tabla, no se precisan cada una de las métricas para los respectivos atributos, por motivos de espacio, pero si se especifica los atributos necesarios para evaluar el producto de software durante la etapa de mantenimiento.

CARACTERÍSTICAS DE LA CALIDAD DEL SOFTWARE SEGÚN NORMA ISO 9126	
Funcionalidad.- Grado en que el software es funcionalmente correcto en correspondencia con las especificaciones y objetivos del usuario para satisfacer los requisitos implícitos y explícitos preestablecidos para el software por el desarrollador y por el usuario respectivamente.	
Subcaracterísticas	Atributos
<i>Compleitud</i>	<ul style="list-style-type: none"> • Configuración total • Contenido total
<i>Consistencia</i>	<ul style="list-style-type: none"> • Automatización • Uniformidad de la estructura, del contenido y del formato de los componentes del software. • Uniformidad del vocabulario, de la simbología y de otras convenciones utilizadas. • Uniformidad de retorno al procesamiento
<i>Corrección</i>	<ul style="list-style-type: none"> • Utilización correcta del idioma español • Correspondencia de las descripciones con los objetos • Funcionamiento correcto
<i>Integridad</i>	<ul style="list-style-type: none"> • Auditabilidad • Seguridad • Autochequeo
<i>Normalización</i>	<ul style="list-style-type: none"> • Cumplimiento de las normas de programación • Cumplimiento de las normas de vocabulario • Cumplimiento de las normas de simbología • Cumplimiento de las normas de aseguramiento de la calidad • Cumplimiento de las normas específicas de la actividad objeto de investigación
Confiabilidad.- Capacidad del software de mantener el nivel de ejecución bajo un conjunto de condiciones previamente establecidas con la precisión requerida durante un período de tiempo de ejecución determinado.	
Subcaracterísticas	Atributos
<i>Exactitud</i>	<ul style="list-style-type: none"> • Exactitud de los cálculos
<i>Recuperabilidad</i>	<ul style="list-style-type: none"> • Opciones de recuperabilidad
<i>Tolerancia de errores o fallos</i>	<ul style="list-style-type: none"> • Verificación de la memoria interna y externa (para la instalación del producto, la reconfiguración del producto o para la salva de la información) • Validación previa de condiciones potenciales de errores (particiones de trabajo inexistente, disponibilidad e integridad de los ficheros, disponibilidad de periféricos) • Tratamiento de los errores (detección y corrección de errores internos del software) • Procesamiento degradado (procedimientos para el funcionamiento degradado en caso de fallos no recuperables como la ausencia de ficheros o deficiencias del hardware)
Mantenibilidad.- Grado de facilidad que brinda el software para que pueda ser mantenido mediante un mantenimiento correctivo, perfectivo o adaptativo. El software debe ser fácilmente mantenible por los programadores de mantenimiento y especialistas de ingeniería del software.	

Subcaracterísticas	Atributos
<i>Autodocumentación</i>	<ul style="list-style-type: none"> Documentación interna
<i>Claridad</i>	<ul style="list-style-type: none"> Claridad de los mensajes Claridad de los nombres de objetos (programas, pantallas, módulos, funciones, etc.)
<i>Concisión</i>	<ul style="list-style-type: none"> Programación procedimental (buenas prácticas de programación)
<i>Diagnosticabilidad</i>	<ul style="list-style-type: none"> Localización del fallo o error (información suficiente para localizar y corregir la causa del error)

Tabla 1. Características, subcaracterísticas y atributos de la calidad del software

3.2 Proceso de Agregación (Sistema de Inferencia Neurodifusa)

Una vez obtenido los valores de medida para cada uno de los atributos definidos para la etapa de mantenimiento, se inicia un proceso de agregación, como se muestra en la figura 1. Dicho proceso de agregación se basa en un sistema de inferencia neurodifuso que resulta de la combinación de dos técnicas: la lógica difusa y las redes neuronales.

La aplicación más extendida de la lógica difusa es sin duda un sistema de inferencia difusa, el único sistema capaz de tratar variables numéricas con variables lingüísticas de modo formal [4],[10]. Básicamente un sistema de inferencia difuso está compuesto por cinco bloques o componentes funcionales, como se presenta en la figura 2.

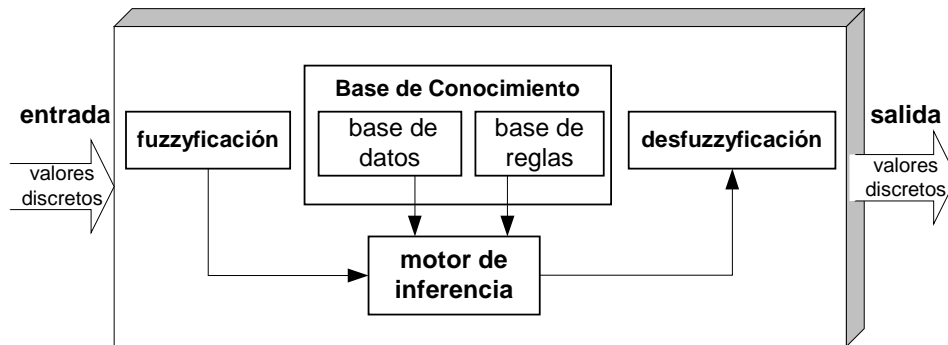


Figura 2. Sistema de inferencia neurodifuso

El presente sistema de inferencia difusa, utiliza un modelo de red neuronal asociativa para almacenar cada una de las reglas difusas. El proceso de inferencia es llevado a cabo por el conjunto de redes, que conforman el motor de inferencia. A continuación se explica cada uno de los componentes:

3.2.1 Fuzzyficación

La fuzzyficación es el proceso de transformación de un valor discreto (crisp) a un valor difuso. Para llevar a cabo dicho proceso, es necesario definir : variables lingüísticas, valores o etiquetas lingüísticas, funciones de membresía, y un dominio de valores discretizado [13]. Para la evaluación de la calidad del software en la etapa de mantenimiento, se definieron:

<i>Variables lingüísticas</i>	<i>Valores lingüísticos</i>	<i>Funciones de membresía</i>	<i>Dominio de valores discretos</i>
1) Funcionalidad 2) Confiabilidad 3) Mantenibilidad 4) Calidad del software	Malo Regular Bueno Muy Bueno	Forma triangular y trapezoidal	Valores reales entre 0 y 3.(Dicho dominio de valores son los mismos para todas las variables lingüísticas definidas).

Tabla 2. Elementos principales para el proceso de fuzzyficación

La Figura 3 representa de manera gráfica las funciones de membresía para las tres primeras variables lingüísticas.

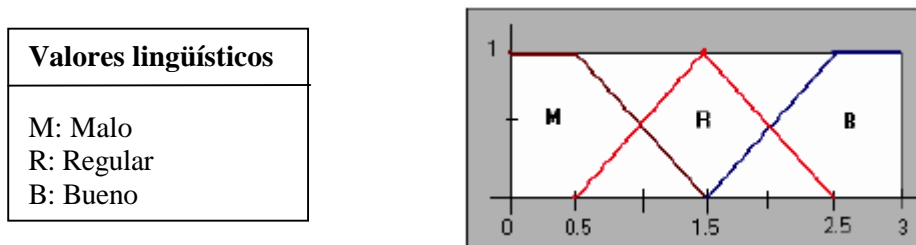


Figura 3. Funciones de Pertenencia. (Antecedente)

Las ecuaciones paramétricas de las funciones de membresía definidas son:

$$\mu_M(x) = \begin{cases} 1 & x \geq 0 \text{ y } x \leq \frac{1}{2} \\ -x + \frac{3}{2} & x > \frac{1}{2} \text{ y } x < \frac{3}{2} \end{cases} \quad \text{ecuación (1)}$$

$$\mu_R(x) = \begin{cases} x - \frac{1}{2} & x \geq \frac{1}{2} \text{ y } x \leq \frac{3}{2} \\ -x + \frac{5}{2} & x > \frac{3}{2} \text{ y } x < \frac{5}{2} \end{cases} \quad \text{ecuación (2)}$$

$$\mu_B(x) = \begin{cases} x - \frac{3}{2} & x \geq \frac{3}{2} \text{ y } x < \frac{5}{2} \\ 1 & x \geq \frac{5}{2} \text{ y } x \leq 3 \end{cases} \quad \text{ecuación (3)}$$

Con la finalidad de aumentar la flexibilidad y la certeza en la evaluación de la calidad, se definió cuatro valores lingüísticos que representan a los conjuntos difusos: Muy bueno, Bueno, Regular y Malo, para la cuarta variable lingüística. (Vea figura 4).

Valores lingüísticos
M: Malo
R: Regular
B: Bueno
MB: Muy Bueno

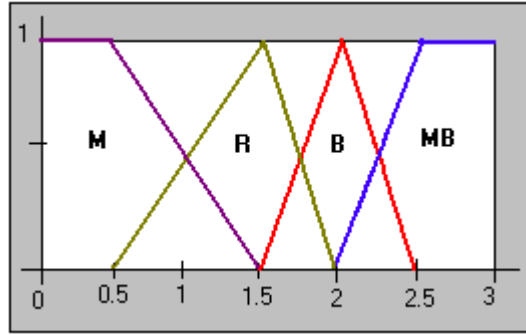


Figura 4 Funciones de Pertenencia. (Consecuente)

De igual manera las ecuaciones paramétricas de dichas funciones de membresía son:

$$\mu_M(x) = \begin{cases} 1 & x \geq 0 \text{ y } x \leq \frac{1}{2} \\ -x + \frac{3}{2} & x > \frac{1}{2} \text{ y } x < \frac{3}{2} \end{cases} \quad \text{ecuación (4)}$$

$$\mu_R(x) = \begin{cases} x - \frac{1}{2} & x \geq \frac{1}{2} \text{ y } x \leq \frac{3}{2} \\ -2x + 4 & x > \frac{3}{2} \text{ y } x < 2 \end{cases} \quad \text{ecuación (5)}$$

$$\mu_B(x) = \begin{cases} 2x - 3 & x \geq \frac{3}{2} \text{ y } x < 2 \\ -2x + 5 & x \geq 2 \text{ y } x \leq \frac{5}{2} \end{cases} \quad \text{ecuación (6)}$$

$$\mu_{MB}(x) = \begin{cases} 2x - 4 & x \geq 2 \text{ y } x < \frac{5}{2} \\ 1 & x \geq \frac{5}{2} \text{ y } x \leq 3 \end{cases} \quad \text{ecuación (7)}$$

3.2.2 Inferencia

Para llevar a cabo el proceso de inferencia es importante definir :

Una *base de reglas* que contiene un número de reglas difusas IF-THEN, que se encargan de relacionar dos o más afirmaciones difusas mediante los operadores OR o AND. Las reglas difusas definidas en el presente trabajo son del siguiente tipo:

Si m es A y n es B y p es C entonces q es D

Cada regla definida está compuesta por tres antecedentes, conectados por el operador lógico “Y”. La simbología utilizada para la representación de las variables y valores lingüísticos (malo, regular, bueno y muy bueno) se muestran en la tabla 3:

Variables Lingüísticas	Mala	Regular	Buena	Muy Buena
Funcionalidad (x1)	A1	A2	A3	
Confiabilidad (x2)	B1	B2	B3	
Mantenibilidad (x3)	C1	C2	C3	
Calidad (y)	Q1	Q2	Q3	Q4

Tabla 3. Variables y valores lingüísticos

Además de determinar una base de reglas difusas, el **motor de inferencia** es otro componente fundamental. El modelo de red neuronal difusa conocida como FAM (Fuzzy Associative Memory), propuesto por B. Kosko[7], es utilizado como memoria asociativa para el almacenamiento y evaluación de las reglas difusas. Dicho motor de inferencia está formado por 14 redes FAM_i que representan el número de reglas difusas determinadas. Cada una de estas redes está conformada por 3 subredes FAM_{AiQj} que representa el número de antecedentes por regla. Finalmente, cada subred está compuesta por 7 neuronas en la capa de entrada y 11 neuronas en la capa de salida que representan el número de valores discretos a tomarse en cuenta para cada una de las variables lingüísticas (x1, x2, x3, y)

Con la finalidad de presentar una figura menos compleja, a continuación se esquematiza la estructura del motor de inferencia con sólo cuatro reglas difusas. (Vea figura 5).

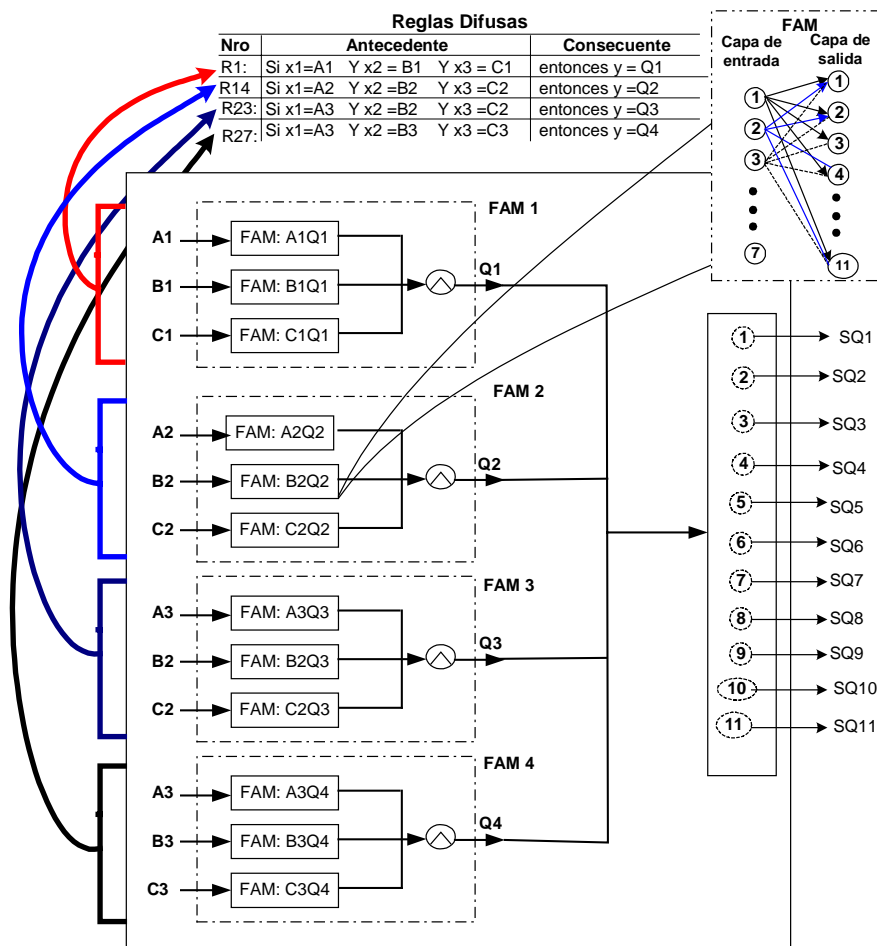


Figura 5. Esquema del motor de inferencia

Para la evaluación de la reglas difusas, es necesario considerar 2 aspectos:

1.- *Ponderar las reglas* que van a gobernar el funcionamiento del sistema. En vista que nos basamos en un modelo cualimétrico válido para todo tipo de software, el valor de los pesos para las reglas es de 1 por considerarse a todas de igual importancia.

2.- *Adicionar una última capa (SUM)* de 7 neuronas equivalente al número de neuronas de la capa de salida de una subred FAM. Como se muestra en la figura 6, los valores de los subconjuntos difusos Q_i extraídos de cada una de las redes FAMi forman la entrada de la capa SUM, encargada de realizar la suma de dichos valores para obtener un conjunto global difuso, que represente el nivel de calidad del software.

En cuanto a los vectores de entrada para la red (E_A , E_B y E_C) se obtienen de la siguiente manera: Es 1 para el valor “más cercano” al valor de la entrada y las demás componentes tienen el valor de 0. Este concepto de “mas cercano” implica pasar por el módulo de fuzzyficación para obtener valores de pertenencia a un determinado conjunto difuso.

3.2.3 Desfuzzyficación

Existen una variedad de métodos para el proceso de fuzzyficación, el método seleccionado es el del centro _ máximo [12], que resulta de la combinación de dos métodos, conocidos como el método del máximo y el método del centroide.

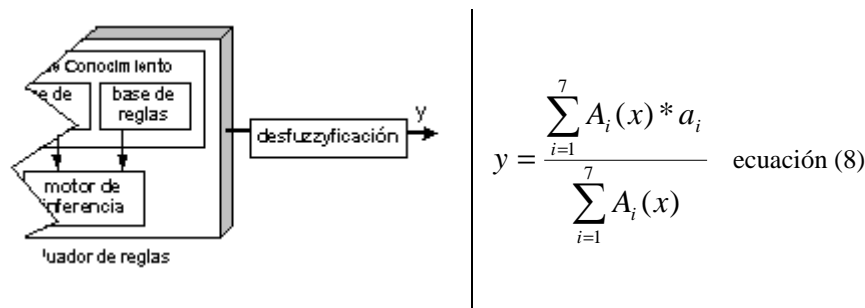


Figura 8. Proceso de desfuzzyficación

Donde:

$A_i(x)$es el grado de pertenencia(altura) de la muestra i-ésima del rango de valores discretos.

a_ies el valor del dominio donde se encuentra el pico de la función de pertenencia.

Por lo tanto, el valor desfuzzyficado representa el nivel de calidad del software para la etapa de mantenimiento. Dicho valor es almacenado en una base de datos para facilitar el seguimiento en la calidad del software para cada proceso de auditoria.

4 Conclusiones

- La determinación del número de reglas fue realizada en base a criterios netamente experimentales, para ello se apoyo en un motor de inferencia elaborado en C++, y mediante pruebas de ensayo y error se determinó 7 reglas como mínimo para un aceptable performace.
- La arquitectura del sistema propuesto para la evaluación de la calidad del software, mantiene su estructura y es adaptable al modelo de evaluación que se elija, pues se aprovecha la capacidad de genericidad y aprendizaje de las redes neuronales.
- Es necesario considerar un determinado dominio para evaluar un tipo específico de software y así poder determinar de manera adecuada el peso respectivo para cada una de las reglas difusas definidas.

- Debido a la diversidad de criterios de evaluación existente por parte de los especialistas de la ingeniería del software, es conveniente mejorar el ajuste o refinado de las funciones de membresía a los conjuntos difusos definidos para la evaluación de la calidad del software.
- Esta dentro de nuestros planes, elaborar una herramienta inteligente de soporte al auditor del software, con la finalidad de mejorar la calidad de dicho producto.

Referencias

- [1] Boehm, B.; Brown, J.R.; Kaspar, J.R., et al. 1978, *Characteristics of Software Quality*; TRW Series of Software Technology.
- [2] Condori-Fernández N.; Sistema Híbrido Neurodifuso para la Evaluación de la Calidad del Software en un Proceso de Auditoría Informática; Actas de las Jornadas Chilenas de Ciencias de la Computación II Workshop de Inteligencia Artificial; Punta Arenas; Noviembre-2001.
- [3] Condori-Fernández N.; Modelo de Evaluación de Especificación de Requisitos de Software Basado en un Sistema de Inferencia Difuso. Memorias del 5to Workshop de Ingeniería de Requisitos y Desarrollo de Ambientes Software, IDEAS, La Habana-Cuba, Abril-2002.
- [4] Espinoza J. y Vandewalle J.; "Técnicas neurodifusas. Una alternativa aplicable al control automático; 1995.
- [5] Galvão L. y Antunes A.; Um Modelo de Avaliação de Especificações Semi-Formais de Requisitos de Software Baseado na Teoria dos Conjuntos Nebulosos; Actas IV Workshop en Requisitos, Buenos Aires 2001.
- [6] Gutiérrez A.; Metodología para el Aseguramiento de la Calidad del Software; Instituto Politécnico Nacional; México, 2000.
- [7] Hilera J.; Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones; Addison-Wesley Iberoamericana. 1995, pp 322-368.
- [8] IEEE Std 1061-1992; *IEEE Standard for a Software Quality Metrics Methodology*; IEEE Computer Society Press
- [9] ISO/IEC 9126:1991; Information technology - Software product evaluation - Quality characteristics and guidelines for their use.
- [10] Jyh-Shing R.; Adaptive-Network-Based Fuzzy Inference System.
- [11] McCall, J.A; Richards, P.K.; Walters, G.F.; 1977, *Factors in Software Quality*; RADC TR-77-369.
- [12] Martín J.; Implementación de redes neurodifusas para ser aplicadas en problemas de clasificación y modelización; USA 2000, <http://www.dissertation.com/lybrary/112113xa.htm>.
- [13] Ojala T.; Neuro-Fuzzy Systems In Control; Master Thesis: Electrical Engineering; Tampere, Finland; 1994.
- [14] Ruiz F.; Mantenimiento del Software; Universidad de Castilla- La Mancha; España, 1999.