Modelo de calidad basado en lógica difusa para identificar microservicios con baja mantenibilidad

Marta de Castro Leira, Elena Goiria Anzuela
Universidad de Salamanca
Salamanca

2024

Resumen

La Arquitectura de Microservicios es un método de desarrollo de aplicaciones software, que consiste en construir una aplicación como un conjunto de pequeños servicios que se ejecutan de manera independiente, y su mantenibilidad es crucial para verificar cómo de fácil es hacer cambios y comprobar que el sistema siga funcionando correctamente después de ellos. Debido a la ambiguedad de algunos aspectos de calidad de software, el modelo que se utiliza en este artículo utiliza lógica difusa. Para ello, se utiliza ISO/IEC 250xy, para convertir datos concretos en niveles difusos evaluando características como la modificabilidad y la comprobabilidad. El sistema genera una puntuación que indica si un sistema necesita mejoras. Se realizaron pruebas en las cuales se obtuvo que microservicios necesitaban ser reemplazados, con alta precisión.

1. Introducción

En los últimos años, el concepto de 'microservicios' ha ganado popularidad en la industria del software por sus beneficios en productividad, permitiendo el desarrollo de aplicaciones como una colección de servicios independientes que se comunican mediante protocolos ligeros. La Arquitectura de Microservicios facilita la mantenibilidad y escalabilidad del software, aunque el mantenimiento de numerosos servicios puede ser un desafío. Es esencial evaluar la calidad de cada microservicio y realizar reajustes para asegurar la confiabilidad y escalabilidad del sistema. A pesar de los métodos existentes, se necesita una metodología de evaluación de calidad específica para la Arquitectura de Microservicios.

Este artículo es una reescritura del trabajo original de Rahime Yilmaz y Feza Buzluca titulado 'A fuzzy logic-based quality model for identifying microservices with low maintainability', publicado en 'Journal of Systems and Software' en 2024. [9] Si bien se mantiene el enfoque general sobre la evaluación de la mantenibilidad en arquitecturas de microservicios, se han introducido cambios.

Con el fin de abordar este problema, se propone un modelo jerárquico de calidad basado en lógica difusa que utiliza métricas para medir y evaluar la mantenibilidad de arquitecturas de microservicios. El objetivo principal de el método es ayudar a los desarrolladores a identificar los microservicios con bajos niveles de mantenibilidad que requieren refactorización.

Las características de alto nivel, como la mantenibilidad, generalmente se evalúan una vez que el sistema ha sido completado y desplegado, ya que es necesario observar su rendimiento en el entorno operativo. Sin embargo, el modelo jerárquico conecta la mantenibilidad con propiedades cuantificables de bajo nivel del software, permitiendo predecir la mantenibilidad durante el proceso de desarrollo. Esto sigue el estándar ISO/IEC 25010 [5], centrándose en dos subcaracterísticas principales: **modificabilidad** y **comprobabilidad**.

Las métricas de software proporcionan información valiosa, pero interpretar sus valores de manera categórica puede ser complicado, ya que un valor puede caer en áreas grises entre diferentes categorías (por ejemplo, entre bajo y medio). Para abordar este problema, se utiliza un sistema basado en lógica difusa, que permite que un valor pertenezca a múltiples categorías simultáneamente. Esto permite evaluar la modificabilidad y comprobabilidad de los microservicios utilizando

reglas de inferencia formuladas en lenguaje natural.

Tras aplicar estas reglas, se utiliza un método de defuzzificación para convertir los resultados difusos en valores numéricos precisos, lo que permite calcular la puntuación de mantenibilidad para cada microservicio evaluado.

Para validar este enfoque, se creó un conjunto de prueba de microservicios con la ayuda de tres desarrolladores, quienes evaluaron y categorizaron 36 microservicios del proyecto de código abierto *Train Ticket*. El modelo propuesto identificó correctamente todos los microservicios etiquetados como BAJOS, alcanzando una precisión del 100 %. Además, los resultados obtenidos fueron coherentes con las etiquetas de nivel medio y alto, lo que indica que el método es efectivo para evaluar la calidad de los microservicios en términos de mantenibilidad.

La mantenibilidad de los microservicios es un tema de interés en la investigación sobre la calidad de la Arquitectura de Microservicios. Bogner et al. (2017) propusieron un modelo de calidad basado en métricas para evaluar la mantenibilidad de sistemas basados en microservicios, definiendo la mantenibilidad como un conjunto de propiedades medidas por métricas específicas. [2] Aunque su modelo es jerárquico y abarca atributos de mantenibilidad, no se había evaluado completamente en su publicación. En contraste, este modelo utiliza un sistema de lógica difusa para medir la mantenibilidad de microservicios individuales y se basa en el proyecto de código abierto *Train Ticket* para evaluar su rendimiento. Otros estudios, como el de Apel et al. (2019), exploraron métricas para evaluar características de calidad en arquitecturas de microservicios, [1] y Cardelli et al. (2019) presentaron MicroQuality, una aproximación para la evaluación de la calidad en Arquitectura de Microservicios.[3] También se examinaron las métricas de calidad en estudios recientes. Finalmente, en un trabajo previo (Yilmaz y Buzluca, 2021), se propuso un modelo jerárquico preliminar para evaluar la mantenibilidad de microservicios basado en lógica difusa básica, que carecía de un proceso de defuzzificación. [8] Ahora, en este estudio, el modelo ha sido mejorado incorporando funciones triangulares y trapezoidales para gestionar áreas ambiguas y se ha añadido un proceso de defuzzificación que permite asignar valores precisos a las subcaracterísticas de mantenibilidad.

El resto del artículo se organiza de la siguiente manera: La Sección 2 ofrece una visión general del modelo jerárquico de calidad y la implementación del sistema de lógica difusa, y describe el modelo propuesto y el método utilizado para evaluar la mantenibilidad. La Sección 3 presenta el problema a resolver y los experimentos a realizar. La Sección 4 presenta los resultados experimentales obtenidos, su validación y discute los hallazgos obtenidos. Por último, la Sección 5 concluye el estudio y propone futuras líneas de investigación.

2. Modelos/Técnicas aplicadas

En este apartado se presenta la metodología utilizada para evaluar con precisión la mantenibilidad de los microservicios. Para ello, se va a explorar elementos que influyen en nuestra metodología, relacionados con los modelos jerárquicos de calidad y la aplicación de la lógica difusa.

En la figura 1 se construye un sistema de medición basado en lógica difusa para establecer las conexiones cuantitativas entre las capas de el modelo jerárquico, el cual vincula la mantenibilidad de las características de calidad externas (de alto nivel) de los microservicios con sus propiedades internas (de bajo nivel). Primero, se lleva a cabo un proceso de fuzzificación en los valores de las métricas para obtener elementos de medición de calidad (QME) para cada propiedad como grados de membresía a conjuntos difusos en tres categorías: BAJO, MEDIO y ALTO. En segundo lugar, utilizando una función de medición basada en las reglas de inferencia y técnicas de defuzzificación, los QME se combinan para obtener medidas de calidad (QM) como medidas cuantitativas de las subcaracterísticas modificabilidad (QM1) y comprobabilidad (QM2). Finalmente, se calcula el promedio ponderado de estas subcaracterísticas para obtener el QM de la característica principal de calidad, mantenibilidad, para cada microservicio. Este QM se utiliza para decidir si un microservicio requiere mejora. Todos estos conceptos se explicaran en profundidad más adelante.

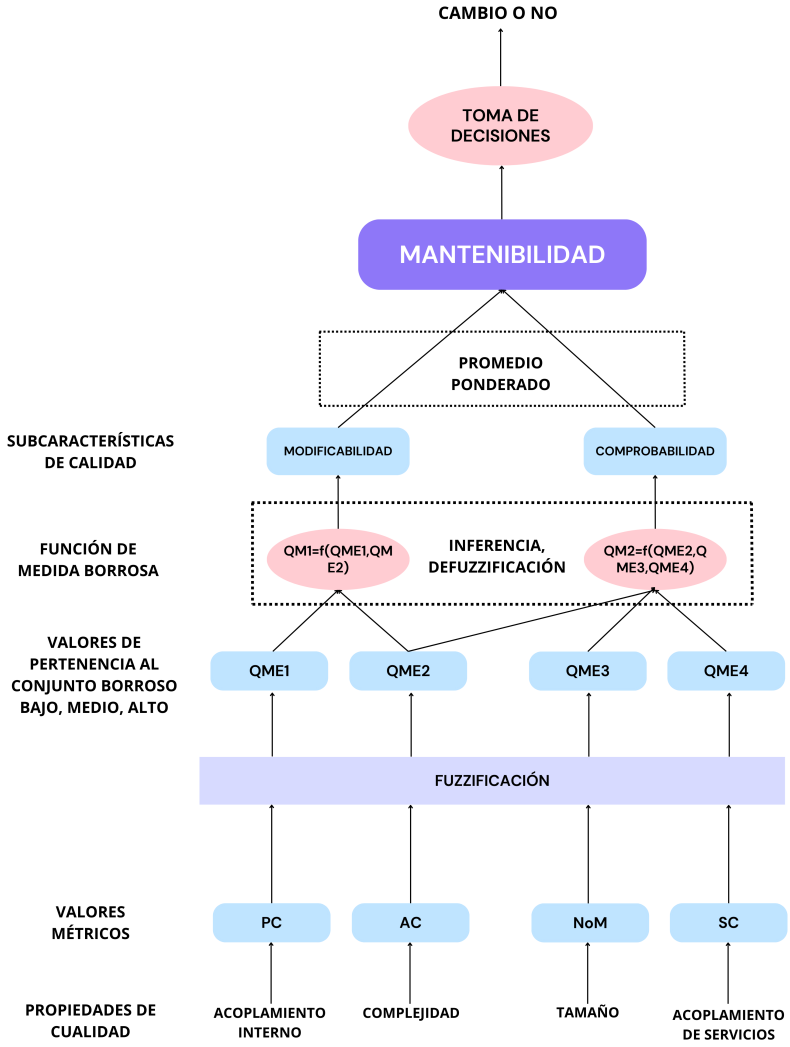

Figura 1: Modelo de calidad jerárquico propuesto para Arquitectura de Microservicios.

2.1. Modelo jerárquico de la calidad

Es necesario evaluar las características de alto nivel de los sistemas de software. Sin embargo, la medición directa de las características externas de alto nivel del software solo es posible una vez que el sistema se ha completado. Además, las propiedades de bajo nivel dependen del diseño y la codificación del sistema, y pueden ser directamente evaluadas durante el proceso de desarrollo. Por tanto, se utiliza los modelos jerárquicos de calidad, los cuales determinan las propiedades de bajo nivel de los sistemas de software que afectan a las propiedades de alto nivel.

ISO/IEC 25010 [5] es un estándar internacional que define cómo medir y evaluar la calidad de un software, basandose en un conjunto de términos y un modelo jerárquico que clasifica la calidad del producto en características específicas, que a su vez se dividen en subcaracterísticas más detalladas. La mantenibilidad es una de las principales características de este modelo de calidad del producto, y se compone de cinco subcaracterísticas: modularidad, reutilización, analizabilidad, modificabilidad y comprobabilidad. el estudio sigue esta norma y evalúa la mantenibilidad de los microservicios, enfocándose en dos subcaracterísticas: modificabilidad y comprobabilidad.

El enfoque de medición se basa en un modelo llamado 'Modelo de Referencia de Medición de Calidad' de la norma ISO/IEC 25020. [6] Esta norma describe cómo medir las características de calidad de alto nivel en función de las de bajo nivel, es decir, proporciona el modelo de referencia para la medición de la calidad del software, explicando las técnicas, métodos y herramientas que se pueden utilizar para obtener valores numéricos de las propiedades medibles del software.

Las propiedades de calidad (QP) medibles de un sistema o producto de software se conocen como propiedades cuantificables y pueden relacionarse con medidas de calidad. Estas propiedades se miden aplicando un método de medición. El valor que se obtiene de este proceso se llama elemento de medida de calidad (QME). Los QME se combinan correctamente a través de una función de medición, lo que genera medidas de calidad (QM), las cuales cuantifican las subcaracterísticas y características de calidad que se desean evaluar.

Las normas ISO proporcionan un marco que permite a los profesionales del software mejorar la comunicación y comprensión al evaluar la calidad. No obstante, estas normas no indican de manera específica qué propiedades del software deben ser cuantificadas para llevar a cabo esa medición. Por esta razón, se han desarrollado un conjunto de métricas específicas de software estático, las cuales se enfocan en evaluar aspectos clave como la modificabilidad y la comprobabilidad. Ahora, se explica detalladamente dichas características para poder entender mejor su función.

Modificabilidad: En los microservicios, la modificabilidad es un atributo clave de la calidad, ya que permite que el sistema evolucione con el tiempo para adaptarse a nuevas funcionalidades, responder a las necesidades cambiantes del negocio y corregir errores sin necesidad de rehacer todo el sistema. Además, la modificabilidad está vinculada a otras subcaracterísticas como la modularidad, reutilización y analizabilidad. Para que un microservicio sea modificable, debe contar con un diseño interno flexible y fácil de entender, con dependencias mínimas. Esto facilita la implementación de cambios sin generar errores o comprometer la calidad del sistema existente.

Comprobabilidad: La comprobabilidad es la propiedad que evalúa en qué medida un sistema, componente o unidad puede ser probado. En el contexto de los microservicios, esta capacidad permite probar de forma individual cada microservicio, ya sea de manera aislada o en interacción con otros servicios. Por tanto, esta propiedad es esencial, ya que para mejorar la calidad y la fiabilidad global del sistema, es fundamental asegurar que los microservicios sean fácilmente comprobables.

2.2. Propiedades de calidad

Como ya se ha determinado anteriormente, las propiedades de calidad internas de bajo nivel son dimensiones fundamentales para valorar y evaluar la calidad de los sistemas de software. Sin embargo, el uso de demasiadas características hace que el modelo sea incomprensible y no aumente significativamente la precisión de la medición. Por tanto, en este estudio, se examinan cuatro aspectos clave de calidad de los microservicios que influyen en su capacidad de modificabilidad y comprobabilidad: acoplamiento interno, acoplamiento de servicios, tamaño y complejidad. A continuación se muestra en detalle la explicación de las propiedades de calidad y sus relaciones con las características de calidad de alto nivel.

Acoplamiento interno: Esta propiedad se refiere al grado en que diferentes componentes dependen o interactúan unas con otras. Además, se considera un indicador de la cohesión de un microservicio. Es común que se trate de un grado alto, ya que un microservicio se compone de clases relacionadas y cooperantes. Sin embargo, un alto grado de acoplamiento interno puede reducir la modificabilidad de los microservicios en el modelo, debido a que si el grado de acomplamiento interno aumenta demasiado, esto es, si la dependencia entre clases es demasiado alta, también aumenta el esfuerzo necesario para modificar un microservicio.

Acoplamiento de servicios: Esta propiedad se refiere al grado en que diferentes servicios dependen o interactúan unos con otros. Cuando un servicio necesita utilizar la funcionalidad de otro servicio, realiza una llamada a dicho servicio. De esta manera, existe una dependencia entre ambos servicios, lo que contradice uno de los objetivos del diseño de una arquitectura de microservicios: crear servicios débilmente acoplados para hacerlos más independientes y más fáciles de probar y escalar. Por tanto, si el servicio de acoplamiento de un microservicio es alto, es difícil entenderlo y probarlo independientemente de otros servicios, lo que genera que reduzca la capacidad de comprobabilidad.

Tamaño: Esta propiedad impacta directamente en los beneficios esperados, por lo que es muy

importante en el diseño de una arquitectura de microservicio. Es necesario encontrar un equilibrio entre la cantidad de funciones que puede realizar y la facilidad con la que se puede probar, para poder determinar el tamaño óptimo para los microservicios. El equilibrio adecuado permite mejorar los procesos de desarrollo, prueba e implementación, lo que genera microservicios más eficientes. Si un microservicio tiene muy pocas funciones puede resultar en una subutilización de sus capacidades, mientras que un microservicio con demasiadas funciones puede exigir un mayor esfuerzo de prueba, comsumiendo muchos recursos y tiempo, y puede presentar dificultades, lo que potencialmente resulta en una cobertura de prueba incompleta.

Complejidad: Esta propiedad se refiere a la métrica de métodos ponderados por clase (WMC, siglas en inglés).[4] En este estudio, se utiliza la métrica AMC (siglas en inglés), la cual es una variante de la métrica WMC, y estudia la complejidad promedio de los métodos.[10] En el desarrollo de software, la complejidad se considera una de las propiedades de calidad más importantes, ya que una alta complejidad afecta negativamente tanto a la modificabilidad como a la comprobabilidad.

Para cuantificar las propiedades de calidad de los microservicios, se analizan varias métricas publicadas basadas en servicios. Se utiliza el coste de propagación (PC) para el acoplamiento interno, la llamada de servicio (SC) para el acoplamiento de servicios, la cantidad de métodos (NoM) para el tamaño y la complejidad promedio (AC) para la complejidad. (Todas las siglas hacen referencia a sus nombres en inglés). A continuación, se ofrecen explicaciones detalladas de estas métricas:

Coste de propagación (PC): Esta métrica mide el impacto que tiene un cambio en una componente sobre otros elementos que dependen de ella. Varía según la cantidad de clases de un microservicio y las dependencias e interacciones entre ellas. Se calcula dividiendo la dependencia promedio del componente por el número total de elementos en el microservicio. En este estudio, un alto coste indica un grado de acoplamiento interno alto.

Relación de llamadas de servicio (SC): Para calcular esta métrica, se divide la cantidad de llamadas realizadas por un microservicio por la cantidad total de microservicios en el proyecto, lo que la hace independiente del tamaño del proyecto.

Número de métodos (NoM): Esta métrica se refiere a la cantidad de funciones u operaciones distintas implementadas dentro de un microservicio.

Complejidad promedio (AC): Esta métrica se refiere a la complejidad acumulada de los microservicios.

2.3. Lógica difusa

La lógica difusa es una técnica que se usa en este artículo para hacer más fácil evaluar la calidad de los microservicios cuando no hay respuestas claras. Este tipo de lógico permite en lugar de establecer que algo es verdadero o falso, declarar que algo tiene valores difusos, como por ejemplo, alto, bajo o medio.

2.3.1. Fuzzificación

La fuzzificación es un paso clave en la lógica difusa que ayuda a manejar la incertidumbre en las mediciones de calidad del software. Dado que la calidad del software no puede medirse con umbrales fijos, aspectos como el número de líneas de código o métodos pueden variar en su interpretación. Esto dificulta clasificar los atributos de calidad de forma precisa, ya que pueden existir zonas intermedias.

En un sistema tradicional de toma de decisiones, cada valor métrico solo puede pertenecer a una categoría (BAJO, MEDIO o ALTO), lo que puede llevar a evaluaciones incorrectas para valores cercanos a los límites. En cambio, la fuzzificación permite que los valores pertenezcan simultáneamente a varias categorías con diferentes grados de pertenencia, representando mejor la incertidumbre.

Para calcular los grados de pertenencia en el modelo, se definen funciones de pertenencia que asignan cada métrica (i, como PC, AC, NoM, SC) a tres conjuntos difusos: BAJO, MEDIO y ALTO. Cada función genera un valor entre 0 y 1, indicando el grado de pertenencia de un valor x a cada conjunto. Por ejemplo, $\mu_A(x)$ representa el grado de pertenencia de x al conjunto A.

El resultado de la fuzzificación para cada métrica se expresa como un conjunto de tres elementos que representan los grados de pertenencia a los tres conjuntos difusos:

$$QME_i = \{\mu_i^L(x), \mu_i^M(x), \mu_i^H(x)\}$$

Donde:

- $\mu_i^L(x)$ es el grado de pertenencia al conjunto BAJO,
- $\mu_i^M(x)$ es el grado de pertenencia al conjunto MEDIO,
- $\mu_i^H(x)$ es el grado de pertenencia al conjunto ALTO.

Esto proporciona una evaluación más matizada de la calidad del microservicio, en lugar de una clasificación rígida y única.

Por ejemplo, si la métrica AC de un microservicio tiene un valor $x = 3{,}8$, las funciones de pertenencia podrían generar los siguientes resultados:

$$\mu_{AC}^L(x) = 0{,}5, \quad \mu_{AC}^M(x) = 0{,}5, \quad \mu_{AC}^H(x) = 0$$

Esto significa que x pertenece en un 50 % a las categorías BAJO y MEDIO, y no pertenece a la categoría ALTO.

Los grados de pertenencia para un valor métrico deben sumar 1:

$$\mu_i^L(x) + \mu_i^M(x) + \mu_i^H(x) = 1, \quad \forall i \in \{PC, AC, NoM, SC\}$$

Esto asegura consistencia en la evaluación y hace que los resultados sean interpretables, permitiendo a los desarrolladores tomar decisiones informadas sobre posibles mejoras en los microservicios.

Para definir las funciones de pertenencia en el modelo de calidad, se seleccionan dos formas comunes en los sistemas difusos: funciones trapezoidales para los conjuntos BAJO y ALTO, y funciones triangulares para el conjunto MEDIO. A continuación, se ilustran las funciones de pertenencia en la Figura 2:

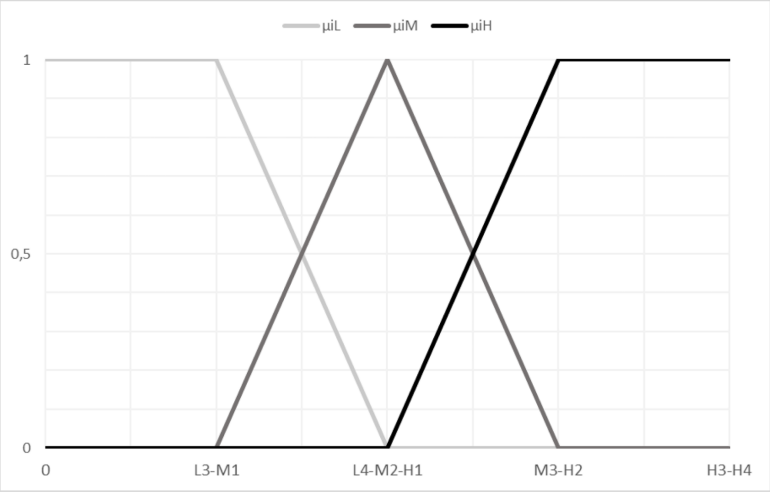

Figura 2: Funciones de pertenencia para las categorías BAJO, MEDIO y ALTO

Función de Pertenencia Trapezoidal para el Conjunto BAJO La función de pertenencia trapezoidal para el conjunto difuso BAJO está determinada por cuatro parámetros clave: L_{1i}, L_{2i}, L_{3i} y L_{4i}. A continuación se describen los puntos clave:

- L_{1i} **y** L_{2i}: Son los puntos donde la función de pertenencia es 1. En este caso, $L_{1i} = L_{2i} = 0$, lo que indica que para $x = 0$, el grado de pertenencia es completamente BAJO para todas las métricas.
- L_{3i} **y** L_{4i}: determinan mediante un análisis estadístico de proyectos de software reales, asegurando que las funciones de pertenencia reflejen con precisión el comportamiento de los microservicios en la práctica.

Ecuación de la función trapezoidal para el conjunto BAJO La función de pertenencia trapezoidal para el conjunto BAJO se expresa matemáticamente como:

$$\mu_i^L(x) = \begin{cases} 1 & \text{si } L_{1i} = L_{2i} = 0 \leq x \leq L_{3i} \\ \frac{L_{4i} - x}{L_{4i} - L_{3i}} & \text{si } L_{3i} < x \leq L_{4i} \\ 0 & \text{si } x > L_{4i} \end{cases}$$

Interpretación de la función de pertenencia La función se interpreta de la siguiente manera:

- **Si** $x \leq L_{3i}$: El grado de pertenencia es 1, lo que significa que el valor se clasifica completamente como BAJO.
- **Si** x **está entre** L_{3i} **y** L_{4i}: El grado de pertenencia disminuye linealmente de 1 a 0, a medida que x se acerca a L_{4i}.
- **Si** $x > L_{4i}$: El grado de pertenencia es 0, lo que indica que el valor no se considera BAJO en absoluto.

Función de Pertenencia Triangular para el Conjunto MEDIO La función de pertenencia triangular se utiliza para definir el conjunto difuso MEDIO. Esta función se caracteriza por tres puntos clave, denominados parámetros: M_{1i}, M_{2i} y M_{3i}. Cada uno de estos parámetros se determina a partir de un análisis estadístico de proyectos de referencia, lo que permite que los valores se ajusten a situaciones reales de microservicios.

- M_{1i}: Este punto representa el límite inferior de la zona que se considera MEDIO.
- M_{2i}: Este es el punto más alto de la función de pertenencia triangular, donde el grado de pertenencia es máximo.
- M_{3i}: Este punto representa el límite superior de la zona MEDIO.

La ecuación para la función de pertenencia triangular es la siguiente:

$$\mu_i^M(x) = \begin{cases} 0 & \text{si } x < M_{1i} \\ \frac{x - M_{1i}}{M_{2i} - M_{1i}} & \text{si } M_{1i} \leq x \leq M_{2i} \\ \frac{M_{3i} - x}{M_{3i} - M_{2i}} & \text{si } M_{2i} < x \leq M_{3i} \\ 0 & \text{si } x > M_{3i} \end{cases}$$

Interpretación de la función de pertenencia La función se interpreta de la siguiente manera:

- **Cuando** $x < M_{1i}$: El grado de pertenencia a la categoría MEDIO es 0, lo que significa que el valor es demasiado bajo para considerarse MEDIO.
- **Cuando** $M_{1i} \leq x \leq M_{2i}$: La función calcula un valor de pertenencia que aumenta linealmente desde 0 hasta 1. Esto significa que, a medida que x se aproxima a M_{2i}, el grado de pertenencia a MEDIO también aumenta.

- **Cuando** $M_{2i} < x \leq M_{3i}$: El grado de pertenencia comienza a disminuir de 1 a 0 a medida que x se acerca a M_{3i}, indicando que el valor ya no puede considerarse MEDIO.
- **Cuando** $x > M_{3i}$: El grado de pertenencia es 0, ya que el valor es demasiado alto para ser clasificado como MEDIO.

Función de Pertenencia Trapezoidal para el Conjunto ALTO La función de pertenencia trapezoidal se utiliza para definir el conjunto difuso ALTO. Esta función se caracteriza por cuatro parámetros clave: H_{1i}, H_{2i}, H_{3i} y H_{4i}. Cada uno de estos parámetros se establece para reflejar cómo se clasifica un valor métrico dentro de la categoría ALTO.

- H_{1i}: Este punto representa el límite inferior donde la pertenencia a la categoría ALTO comienza a aumentar.
- H_{2i}: Este es el punto en el que la pertenencia a la categoría ALTO alcanza su valor máximo (1).
- H_{3i} **y** H_{4i}: Estos puntos representan el límite superior de la función de pertenencia. En el modelo, se establecen en el valor máximo MAX_i de la métrica correspondiente.

Los parámetros H_{1i} y H_{2i} se determinan mediante cálculos estadísticos basados en datos reales de proyectos de referencia, asegurando que las funciones de pertenencia reflejen el comportamiento de los microservicios en situaciones prácticas.

Ecuación de la función trapezoidal para el conjunto ALTO La ecuación para la función de pertenencia trapezoidal del conjunto ALTO se define como:

$$\mu_i^H(x) = \begin{cases} 0 & \text{si } x \leq H_{1i} \\ \frac{x - H_{1i}}{H_{2i} - H_{1i}} & \text{si } H_{1i} < x \leq H_{2i} \\ 1 & \text{si } H_{2i} < x \leq H_{3i} = H_{4i} \end{cases}$$

Interpretación de la función de pertenencia La función se interpreta de la siguiente manera:

- **Si** $x \leq H_{1i}$: El grado de pertenencia a la categoría ALTO es 0, indicando que el valor es demasiado bajo para considerarse ALTO.
- **Si** $H_{1i} < x \leq H_{2i}$: El grado de pertenencia aumenta linealmente desde 0 hasta 1. A medida que x se aproxima a H_{2i}, el valor se considera cada vez más ALTO.
- **Si** $H_{2i} < x \leq H_{3i}$: El grado de pertenencia a la categoría ALTO es 1, lo que significa que el valor se considera completamente ALTO.

Interrelación de Parámetros Al definir las funciones de pertenencia para las categorías de calidad, es fundamental reconocer que algunos parámetros de fuzzificación se superponen. Esto refleja la naturaleza continua y difusa de los valores de calidad, en contraste con las divisiones rígidas de un sistema tradicional.

Los límites entre las categorías se establecen de la siguiente manera:

$$L_{3i} = M_{1i}, \quad L_{4i} = M_{2i} = H_{1i}, \quad \text{y } M_{3i} = H_{2i}$$

La figura 2 ilustra estas funciones de pertenencia, mostrando cómo se solapan y organizan en el espacio de calidad. Esta visualización ayuda a comprender cómo un valor puede estar parcialmente en varias categorías.

La Tabla 1 proporciona un resumen de las variables, los tipos de funciones de pertenencia y los parámetros asociados, facilitando así la interpretación de los resultados del modelo.

Variable Lingüística	Función de Pertenencia	Puntos, Intervalos
BAJO	Trapezoidal	$[L_1 = L_2 = 0, L_3, L_4]$
MEDIO	Triangular	$[M_1, M_2, M_3]$
ALTO	Trapezoidal	$[H_1, H_2, H_3 = H_4 = MAX]$

Cuadro 1: Elementos del proceso de fuzzificación

2.3.2. Determinación de los Parámetros de las Funciones de Pertenencia

El proceso de determinación de los parámetros para las funciones de pertenencia es crucial para representar adecuadamente la incertidumbre y optimizar el rendimiento de los sistemas de lógica difusa.

Para definir los intervalos de las funciones de pertenencia $\mu_i^L(x)$, $\mu_i^M(x)$ y $\mu_i^H(x)$, se analizan los valores métricos de 14 microservicios obtenidos de dos proyectos de referencia: *TeaStore* y *Microservice Observability*. Estos proyectos contienen una gama diversa de microservicios, lo que asegura que las características de calidad varíen y sean representativas.

Análisis de los Valores Métricos En el análisis realizado, se observó que las métricas PC, SC y NoM siguen una distribución normal típica. Para obtener los parámetros de fuzzificación, se calcularon tres estadísticos clave: el primer cuartil (Q1), la mediana (Q2) y el tercer cuartil (Q3), de la siguiente manera:

- **Primer cuartil (Q1)**: Corresponde a los parámetros L_3 y M_1, considerando los valores inferiores a Q1 como BAJOS.
- **Mediana (Q2)**: Corresponde a los parámetros L_4, M_2 y H_1, donde los valores alrededor de la mediana se consideran normales o MEDIOS.
- **Tercer cuartil (Q3)**: Corresponde a los parámetros M_3 y H_2, asignando los valores superiores a Q3 a la categoría ALTO.

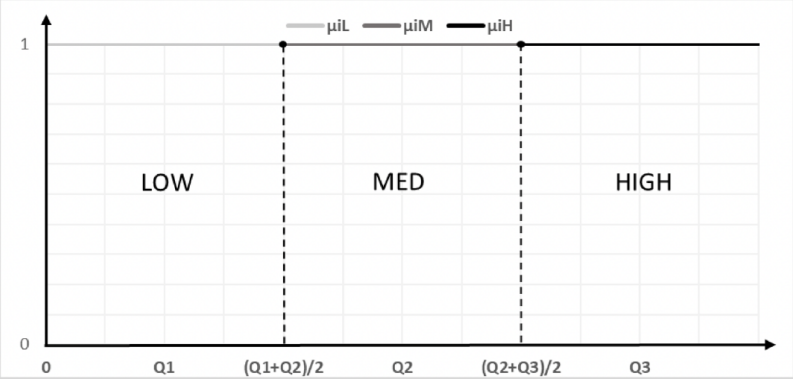

Figura 3: División en cuartiles

En la figura 3 se muestra la división en cuartiles del espacio de las funciones de pertenencia.

La Tabla 2 resume los parámetros clave utilizados para definir las funciones de pertenencia de las métricas estudiadas.

Estos parámetros se utilizan para calcular las funciones de pertenencia correspondientes a las métricas, lo que permite clasificar los microservicios en términos de calidad utilizando valores BAJO, MEDIO o ALTO.

Métrica	L1=L2	L3=M1	L4=M2=H1	M3=H2	H3=H4
SC	0	0.125	0.25	0.5	1
PC	0	19.4	24.2	27.1	50
AC	0	2.81	4.78	5.63	11
NoM	0	9	16	30	90

Cuadro 2: Parámetros de las funciones de pertenencia para cada métrica.

Ejemplo de Aplicación de las Funciones de Pertenencia Como ejemplo, se puede observar que un valor de PC menor a 19.4 se clasifica definitivamente como BAJO, con $\mu^L_{PC}(19{,}4) = 1$ y $\mu^M_{PC}(19{,}4) = \mu^H_{PC}(19{,}4) = 0$. En contraste, un valor de 21.8 para la misma métrica tendría grados de pertenencia tanto al conjunto BAJO como MEDIO, con $\mu^L_{PC}(21{,}8) = 0{,}5$ y $\mu^M_{PC}(21{,}8) = 0{,}5$.

2.3.3. Reglas de Inferencia Difusas

En un sistema de lógica difusa, el **Módulo de Inferencia Difusa** aplica reglas del tipo SI-ENTONCES (IF-THEN) a los grados de pertenencia de las entradas, con el fin de obtener resultados difusos en las salidas. En este contexto, la parte SI (antecedente) representa las entradas, mientras que la parte ENTONCES (consecuente) genera las salidas.

Las reglas se construyen a partir de conocimiento experto y los datos empíricos, expresando relaciones entre las propiedades internas de los microservicios, como el acoplamiento y la complejidad, y sus características de calidad, como la modificabilidad. Estas reglas se formulan en términos lingüísticos.

Por ejemplo, una regla podría ser:

> *"Si el acoplamiento interno es alto y la complejidad de un microservicio es alta, entonces su modificabilidad es baja."*

Este tipo de reglas reflejan cómo las características internas de los microservicios afectan directamente a su calidad. Para organizar este proceso de inferencia, se han definido un conjunto de reglas específicas tanto para la modificabilidad como para la comprobabilidad de los microservicios.

La **Tabla 3** presenta 9 reglas de inferencia difusa (RM1 - RM9) utilizadas para determinar el nivel de modificabilidad, mientras que la **Tabla 4** contiene 27 reglas (RT1 - RT27) que determinan los niveles de comprobabilidad de los microservicios, en función de los valores difusos de sus propiedades internas.

Regla	Complejidad (AC)	Tamaño (NoM)	Acoplamiento del Servicio (SC)	Modificabilidad
RM1	MED	HIGH	LOW	LOW
RM2	MED	HIGH	MED	LOW
RM3	MED	HIGH	HIGH	LOW
RM4	HIGH	LOW	HIGH	LOW
RM5	HIGH	MED	LOW	LOW
RM6	HIGH	MED	MED	LOW
RM7	HIGH	MED	HIGH	LOW
RM8	HIGH	HIGH	LOW	LOW
RM9	HIGH	HIGH	MED	LOW

Cuadro 3: Reglas de inferencia difusa: Modificabilidad

Regla	Complejidad (AC)	Tamaño (NoM)	Acoplamiento del Servicio (SC)	Comprobabilidad
RT1	MED	HIGH	LOW	LOW
RT2	MED	HIGH	MED	LOW
RT3	MED	HIGH	HIGH	LOW
RT4	HIGH	LOW	HIGH	LOW
RT5	HIGH	MED	LOW	LOW
RT6	HIGH	MED	MED	LOW
RT7	HIGH	MED	HIGH	LOW
RT8	HIGH	HIGH	LOW	LOW
RT9	HIGH	HIGH	MED	LOW
RT10	HIGH	HIGH	HIGH	LOW
RT11	LOW	LOW	HIGH	MED
RT12	LOW	MED	HIGH	MED
RT13	LOW	HIGH	LOW	MED
RT14	LOW	HIGH	MED	MED
RT15	LOW	HIGH	HIGH	MED
RT16	MED	LOW	HIGH	MED
RT17	MED	MED	LOW	MED
RT18	MED	MED	MED	MED
RT19	MED	MED	HIGH	MED
RT20	HIGH	LOW	LOW	MED
RT21	HIGH	LOW	MED	MED
RT22	LOW	LOW	LOW	HIGH
RT23	LOW	LOW	MED	HIGH
RT24	LOW	MED	LOW	HIGH
RT25	LOW	MED	MED	HIGH
RT26	MED	LOW	LOW	HIGH
RT27	MED	LOW	MED	HIGH

Cuadro 4: Reglas de inferencia difusa: Comprobabilidad

Estas reglas expresan relaciones considerando términos lingüísticos entendibles. Por ejemplo, según la regla RM3 en la **Tabla 3**, si la complejidad de un microservicio es ALTA, entonces su modificabilidad es BAJA. Esta regla refleja que, si sus métodos son complejos, el esfuerzo requerido para modificar el microservicio será elevado, resultando en una modificabilidad baja.

Del mismo modo, la regla RT22 en la **Tabla 4** establece que, si la complejidad, el tamaño y el acoplamiento de un microservicio son todos BAJOS, entonces su comprobabilidad es ALTA. Esto aplica si los métodos de un microservicio son simples, su tamaño es relativamente pequeño, y no está fuertemente acoplado a otros servicios. Dado que será fácil probar un microservicio de este tipo, su nivel de comprobabilidad se considera ALTO.

Las métricas pueden pertenecer a varios conjuntos difusos, permitiendo que un microservicio cumpla múltiples reglas. Para combinar estas reglas de inferencia, se utiliza el **método de inferencia max-min de Mamdani** [7], que consta de dos fases principales:

- **Implicación**: Se calcula la fuerza de cada regla tomando el valor mínimo de los grados de pertenencia de las entradas. Para calcular dicha fuerza de la regla de modificabilidad se utiliza la siguiente fórmula.

$$S_{RMx} = \text{mín}\left(\mu_A^{\text{PC}}(\text{PC}_m), \mu_B^{\text{AC}}(\text{AC}_m)\right)$$
$$A, B \in \{L, M, H\}$$

- **Agregación**: Una vez calculada la implicación de todas las reglas, se toma el valor máximo de los grados de pertenencia calculados. Esto permite obtener una evaluación difusa final sobre la modificabilidad o comprobabilidad.

$$\mu^{\text{Mod}} = \max_x \left(S_{RMx}\right)$$

2.3.4. Defuzzificación

La defuzzificación es la etapa final de un sistema de lógica difusa. Consiste en combinar el conjunto difuso agregado, obtenido del Módulo de Inferencia Difusa, con una función matemática, conocida como método de defuzzificación, para producir un valor numérico nítido. Este valor se

utiliza en cálculos o toma de decisiones, ofreciendo una interpretación clara de los resultados difusos.

En este estudio, se emplea el **método del centroide**, una técnica comúnmente utilizada que calcula el centro de gravedad del conjunto difuso agregado, representando el 'peso total' de las reglas aplicadas. El valor nítido resultante refleja los niveles de modificabilidad y comprobabilidad de los microservicios.

Las funciones de pertenencia son esenciales en el proceso de defuzzificación, ya que permiten convertir los valores difusos en valores numéricos nítidos. Para este propósito, se utilizan las funciones de pertenencia presentadas en la **Tabla 5**, que muestran los parámetros utilizados para evaluar la modificabilidad y comprobabilidad. Estas funciones generan puntuaciones que varían según el nivel de calidad de los microservicios.

Los parámetros seleccionados en este estudio generan puntuaciones en el rango de 17.4 a 82.6, que luego se normaliza en un intervalo de 0 a 100 para facilitar la interpretación. Estas puntuaciones se interpretan de acuerdo con los rangos definidos en la **Tabla 5** y mostrados en la gráfica 4 facilitando la evaluación de la calidad de los microservicios.

Puntuación	**Significado**
0 - 30	BAJO
30 - 40	BAJO – MEDIO, más cercano a BAJO
50	MEDIO
50 - 60	ALTO, más cercano a MEDIO
60 - 70	MEDIO – ALTO, más cercano a ALTO
70 - 100	ALTO

Cuadro 5: Significados de las puntuaciones generadas por el método de defuzzificación.

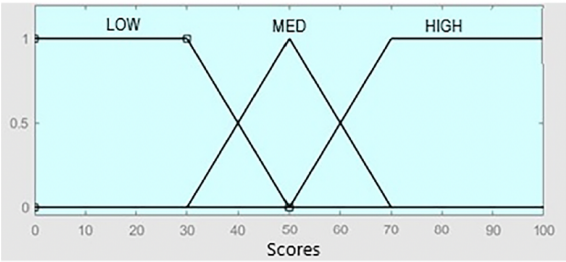

Figura 4: Funciones de pertenencia en la defuzzificación

Ejemplo de aplicación Para ilustrar este proceso, se considera un microservicio con los valores de las métricas $PC = 23{,}4$ y $AC = 6{,}53$. Tras el proceso de fuzzificación, los grados de pertenencia de PC a los conjuntos difusos son los siguientes: BAJO: 0, MEDIO: 0.8, y ALTO: 0.2. Esto se expresa como: $QME_1 = \{0, 0{,}8, 0{,}2\}$. En cuanto al valor de AC, este pertenece únicamente al conjunto ALTO, con un grado de pertenencia de 1, quedando: $QME_2 = \{0, 0, 1\}$. Estos valores activan dos reglas de inferencia difusa: RM6 y RM9. Siguiendo el método de inferencia max-min de Mamdani, se calcula la fuerza de la regla RM6, obteniendo un valor de 0.8, y de la regla RM9, con un valor de 0.2. En la fase de agregación, el resultado final para el conjunto BAJO es 0.8. Finalmente, al aplicar el método del centroide sobre el conjunto difuso agregado, se obtiene una puntuación de $QM_1 = 22{,}8$ para la modificabilidad del microservicio. Dado que este valor está por debajo del umbral de 40, concluyendo que el microservicio tiene un diseño deficiente en términos de modificabilidad.

2.4. Cálculo de la mantenibilidad

Una vez obtenidas las puntuaciones de las subcaracterísticas modificabilidad y comprobabilidad, se calcula su promedio ponderado para determinar el nivel de mantenibilidad de los microservicios. Esto se realiza utilizando la siguiente ecuación:

$$MNT_m = w_{mod} \times MOD_m + w_{tst} \times TST_m$$

- **MNT**$_m$: Representa la puntuación global de mantenibilidad del microservicio m.
- **MOD**$_m$: Es la puntuación de modificabilidad del microservicio m, es decir, la medida de calidad QM_1 obtenida durante la defuzzificación.
- **TST**$_m$: Es la puntuación de comprobabilidad del microservicio m, es decir, la medida de calidad QM_2 obtenida durante la defuzzificación.
- $\mathbf{w}_{mod}$ y $\mathbf{w}_{tst}$: Son los pesos asignados a la modificabilidad y comprobabilidad, respectivamente. Estos valores se encuentran en el rango [0, 1] y su suma siempre es igual a 1, garantizando que la puntuación combinada sea un valor normalizado.

En los experimentos, se asigna un peso de 0.5 tanto a la modificabilidad como a la comprobabilidad ($w_{mod} = w_{tst} = 0{,}5$), dando igual importancia a ambas características en el cálculo de la mantenibilidad del microservicio. Pero las empresas que utilicen este método pueden emplear diferentes pesos dependiendo de la importancia que le asignen a la modificabilidad y comprobabilidad en sus proyectos.

Para decidir si un microservicio necesita refactorización debido a un bajo nivel de mantenibilidad, se compara su puntuación con un umbral predeterminado T_{REF}. Si la puntuación de mantenibilidad es menor o igual al umbral, el microservicio debería considerarse para refactorización. Esto se determina de la siguiente manera:

$$\text{Si } MNT_m \leq T_{REF}, \text{ el microservicio } m \text{ necesita refactorización.}$$

En este estudio, se selcciona un umbral de $T_{REF} = 40$, ya que las funciones de pertenencia que se utilizan en el proceso de defuzzificación generan puntuaciones por debajo de 40 cuando las subcaracterísticas tienden a valores bajos. Sin embargo, las empresas pueden optar por otros valores de umbral.

3. Problema a resolver/Caso de estudio/Experimentos

Se aplica este estudio a un proyecto de código abierto basado en arquitectura de microservicios desarrollado en Java conocido como Train Ticket. Este proyecto se trata de una aplicación utilizada para reserva de billetes de tren, consulta, reserva, pago, cambios y notificaciones al usuario.

Se elige este estudio ya que tiene un tamaño mediano lo cual permite un mejor manejo, y además representa una aplicación de microservicios del mundo real. Sin embargo, no se han utilizado los 41 microservicios que tiene, ya que cinco de ellos no se alinean con los criterios integrales para definir microservicios.

Para obtener mejores resultados, más adelante se comparan los resultados con las opiniones de tres desarrolladores de software que analizaron el proyecto y categorizaron los microservicios con antelación. Se han elegido dichos desarrolladores debido a que cuentan con más de cinco años de experiencia en el desarrollo de sistemas basados en microservicios.

4. Resultados obtenidos y Discusión

La Tabla 6 presenta los resultados del modelo aplicado al proyecto Train Ticket. En la primera columna se asignan números a los microservicios, los cuales están nombrados en la segunda columna. Las siguientes 4 columnas muestran las métricas PC, AC, NoM y SC de los microservicios. Las dos siguientes columnas presentan los valores obtenidos trás realizar la defuzzificación de la modificabilidad y la comprobabilidad. Por último, se muestra el promedio ponderado de las subcaracterísticas en la columna de mantenibilidad.

En el experimento, la modificabilidad y la comprobabilidad tienen los mismos pesos, $w_{mod} = w_{tst} = 0{,}5$. Además, se selecciona $T_{REF} = 40$, por lo que los valores en la columna de mantenibilidad que sean menores o iguales a 40 se mostrarán en negrita, lo que quiere decir, que esos microservicios tienen problemas de mantenimiento y deberían refactorizarse.

MS. Num.	Microservice Name	PC	AC	NoM	SC	Modificabilidad	Comprobabilidad	Mantenibilidad
M1	Servicio de información básica de administración	27	1	67	0.58	50.00	50.00	50.00
M2	Servicio de pedidos de administrador	20.31	3	19	0.14	76.57	70.99	73.78
M3	ts-servicio-de-ruta-admin	26.53	1	17	0.11	57.06	80.27	68.66
M4	ts-servicio-de-viajes-admin	21.53	2.1	23	0.14	77.69	67.75	72.72
M5	ts-servicio-de-usuario-admin	18	1	19	0.17	79.84	74.79	77.31
M6	ts-servicio-de-seguro	17.36	1.97	49	0.25	79.84	50.00	64.92
M7	ts-servicio de autenticación	13.75	4.5	44	0.25	79.25	21.97	50.61
M8	ts-servicio-básico	19.9	4.47	24	0.08	75.58	37.42	56.50
M9	ts-servicio-cancelamiento	10.73	6.27	25	0.08	50.00	18.18	**34.09**
M10	ts-servicio-config.	26.56	1.54	27	0.17	56.67	59.66	58.17
M11	ts-servicio-precio-consigna	26.56	1.58	21	0.14	56.67	71.70	64.18
M12	ts-servicio-consigna	18	1.66	30	0.17	79.84	50.00	64.92
M13	ts-servicio-contactos	23.46	1.52	41	0.22	79.39	50.00	64.70
M14	ts-servicio-ejecutar	23.44	6.53	17	0.08	22.81	17.52	**20.17**
M15	ts-servicio-entrega de comida (mapa)	27.08	4.4	34	0.19	24.72	25.85	**25.29**
M16	ts-servicio-comida	20.14	9.68	39	0.19	40.36	18.39	**29.38**
M17	ts-servicio-pago-interno	12.81	3.98	69	0.25	78.09	29.67	53.88
M18	ts-servicio-notificación	19	1	37	0.14	79.84	50.00	64.92
M19	ts-servicio-pedir otro	18.21	3.96	78	0.44	78.05	29.95	54.00
M20	ts-servicio-pedir	18.21	3.96	84	0.44	78.05	29.95	54.00
M21	ts-servicio-reservar otro pedido	8.22	10.94	23	0.06	50.00	18.53	**34.26**
M22	ts-servicio-reservar pedido	7.91	10.99	23	0.06	50.00	18.53	**34.26**
M23	ts-servicio-precio	22.22	2.06	32	0.17	78.32	50.00	64.16
M24	ts-servicio-reservar de nuevo	11.11	7.37	28	0.08	50.00	17.69	**33.84**
M25	ts-servicio-plan de ruta	16.89	3.17	20	0.11	79.08	65.26	72.17
M26	ts-servicio-ruta	23.46	3.17	34	0.17	75.52	41.42	58.47
M27	ts-servicio-asiento	16.67	5.57	15	0.08	53.48	25.70	**40.00**
M28	ts-servicio-seguridad	16.67	1.64	30	0.17	79.84	50.00	64.92
M29	ts-servicio-estación	26.56	1.9	41	0.25	56.67	50.00	53.34
M30	ts-servicio-infoticket	26.45	1	13	0.08	58.06	81.65	69.86
M31	ts-servicio-train	26.56	1.47	36	0.17	56.67	50.00	53.34
M32	ts-servicio-plan de viaje	14.53	1.91	32	0.14	79.84	50.00	64.92
M33	ts-servicio-viaje	14.37	2.22	66	0.33	79.84	50.00	64.92
M34	ts-servicio-viaje2	14.37	2.62	60	0.33	79.84	50.00	64.92
M35	ts-servicio-usuario	19.83	2.17	38	0.28	79.15	50.00	64.57
M36	ts-servicio-codigo de verificación	25	5.71	16	0.06	21.58	17.37	**19.47**

Cuadro 6: Valores métricos y resultados de los microservicios en la aplicación Train Ticket.

Para validar la precisión de estos hallazgos, se compararon estos resultados a los obtenidos por tres desarrolladores de software experimentados. Evaluaron independientemente la mantenibilidad de cada microservicio, teniendo en cuenta sus criterios de evaluación. Clasificaron los microservicios en tres categorías: BAJA (B), MEDIA (M) y ALTA (A). Los microservicios de la categoría BAJA deben refactorizarse ya que tienen problemas de mantenimiento, los de la categoría MEDIA pueden tener algunos problemas de manteniento aunque tengan partes bien diseñadas. Por último, los microservicios de la categoría ALTA están bien diseñados, por lo que no tienen ningún problema de mantenimiento.

La Tabla 7 presenta las etiquetas asignadas a los microservicios por los investigadores y los valores de mantenibilidad calculados por el método propuesto. Se muestra la misma cantidad de microservicios que en la Tabla 6. La primera columna muestra los microservicios, y las columnas E1, E2 y E3 presentan las etiquetas asignadas (BAJO, MEDIO, ALTO) por los investigadores. La columna "Decisión" muestra la decisión final basada en la opinión mayoritaria de los investigadores.

La última columna contiene los valores de mantenibilidad de los microservicios calculados utilizando el método propuesto.

MS. Num.	E1	E2	E3	Decisión	Mantenibilidad del modelo	MS. Num.	E1	E2	E3	Decisión	Mantenibilidad del modelo
M1	M	A	M	M	50.00	**M19**	M	M	M	M	54.00
M2	A	A	A	A	73.78	**M20**	M	B	M	M	54.00
M3	M	A	A	A	68.66	**M21**	B	B	M	M	**34.26**
M4	A	A	A	A	72.72	**M22**	B	B	M	M	**34.26**
M5	A	A	A	A	77.31	**M23**	M	A	A	A	64.16
M6	M	M	M	M	64.92	**M24**	B	B	B	B	**33.84**
M7	M	A	M	M	50.61	**M25**	A	M	A	A	72.17
M8	M	B	M	M	56.50	**M26**	A	A	M	A	58.47
M9	M	B	B	B	**34.09**	**M27**	M	B	M	M	**39.59**
M10	A	A	A	A	58.17	**M28**	A	A	A	A	64.92
M11	M	A	A	A	64.18	**M29**	A	A	A	A	53.34
M12	M	A	A	A	64.92	**M30**	A	A	A	A	69.86
M13	A	A	M	A	64.70	**M31**	A	A	M	A	53.34
M14	M	M	B	M	**20.17**	**M32**	A	M	A	A	64.92
M15	B	B	B	B	**25.29**	**M33**	M	B	M	M	64.92
M16	B	B	M	B	**29.38**	**M34**	M	M	M	M	64.92
M17	M	M	M	M	53.88	**M35**	M	A	M	M	64.57
M18	A	A	M	A	64.92	**M36**	B	B	M	B	**19.47**

Cuadro 7: Comparación de los resultados del modelo de calidad con la valoración de los evaluadores.

El objetivo de este estudio es identificar microservicios que deben refactorizarse por problemas de mantenimiento. Por tanto, se comparan los 7 microservicios de la categoría BAJO en la Tabla 6: M9, M15, M16, M21, M22, M24 y M36; y los 9 mostrados en negrita en la Tabla 7: M9, M14, M15, M16, M21, M22, M24, M27 y M36. De los 9 microservicios detectados en el estudio, 7 fueron clasificados como BAJO por los investigadores, mientras que los microservicios M14 y M27 fueron clasificados como MEDIO. Por tanto, se realiza una matriz de confusión en la Tabla 8 para presentar el desempeño del método en la detección de microservicios con un nivel de mantenimiento bajo.

Las entradas en la matriz de confusión son:

- TP (Verdadero positivo) es la cantidad de microservicios etiquetados como BAJOS por los evaluadores y predichos correctamente por el método.
- FP (Falso positivo) es la cantidad de microservicios NO etiquetados como BAJOS por los evaluadores y predichos incorrectamente por el método.
- FN (Falso negativo) es la cantidad de microservicios etiquetados como BAJO por los evaluadores y predichos incorrectamente por el método.
- TN (Verdadero negativo) es la cantidad de microservicios NO etiquetados como BAJO por los evaluadores y predichos correctamente por el método.

		Investigadores	
		BAJO	**NO BAJO**
Predecido por el modelo	**BAJO**	TP=7	FP=2
	NO BAJO	FN=0	TN=27
Total		7	29

Cuadro 8: Matriz de confusión.

Para evaluar el desempeño de los modelos de predicción, se calcularon la recuperación, precisión, medida-F y exactitud utilizando los valores de la matriz de confusión en la Tabla 8.

$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP},$$
$$\text{Medida-F} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \quad \text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

En la Tabla 9 se muestran los valores de los resultados obtenidos a partir del experimento en el proyecto Train Ticket.

Recuperación	**Precisión**	**Medida-F**	**Exactitud**
100 %	77.78 %	87.5 %	94.44 %

Cuadro 9: Desempeño del método en la detección de microservicios con bajo nivel de mantenibilidad en el proyecto de Train Ticket.

Los resultados de la Tabla 9 muestran que se puede confiar en el método propuesto para guiar a los desarrolladores en la detección de microservicios con un nivel bajo de mantenibilidad. Los resultados obtenidos muestran que el modelo de lógica difusa detectó con éxito los siete microservicios etiquetados como BAJOS, logrando una puntuación de recuperación perfecta del 100 % . Por otro lado, el modelo clasificó dos microservicios como 'BAJOS', pero los desarrolladores los consideraron 'MED', lo que generó una precisión del $77,78\,\%$, lo que dio como resultado una exactitud del $94,44\,\%$ en la detección de los microservicios etiquetados como BAJOS.

Se analizan los dos microservicios, M14 y M27, que el método categorizó como BAJOS, mientras que los desarrolladores los consideraron MEDIOS. Se observa que uno de los tres desarrolladores también los categorizó como BAJOS. La puntuación calculada de M27 es 39,59, que está cerca del umbral entre BAJO y MEDIO. Dados estos resultados, el modelo identifica con éxito todos los microservicios etiquetados como BAJOS.

En conclusión, los hallazgos indican que el método propuesto puede asignar puntajes de mantenibilidad a microservicios que se correlacionan con las opiniones de los investigadores y detectar microservicios que requieren refactorización. Es decir, los hallazgos indican la posible aplicación de sistemas de medición basados en lógica difusa a varios aspectos de la evaluación de la calidad de los microservicios utilizando modelos de calidad adecuados. El éxito de este modelo en una aplicación ampliamente referenciada como Train Ticket muestra su utilidad práctica para predecir la capacidad de mantenimiento de microservicios individuales durante el desarrollo del sistema de software.

5. Conclusiones y Futuras lineas de trabajo

En este estudio, se propone un nuevo enfoque para medir y evaluar la mantenibilidad de la arquitectura de microservicios utilizando la lógica difusa. Se siguen las normas ISO/IEC 25.010 y 25.020 para crear un modelo jerárquico de calidad que defina las relaciones entre la mantenibilidad, una característica de alto nivel, la modificabilidad y la comprobabilidad, dos subcaracterísticas y las métricas de software relacionadas. Además, el método propuesto aborda la ambigüedad en la interpretación de estas métricas empleando técnicas de lógica difusa, es decir, fuzzificación, inferencia y defuzzificación.

Se evalua la eficacia de este método aplicándolo a un proyecto de código abierto basado en la arquitectura de microservicios, Train Ticket. Se obtienen resultados prometedores que se alinean con las evaluaciones manuales realizadas por desarrolladores experimentados. Estos resultados indican que el modelo propuesto es una herramienta útil para los equipos de desarrollo, ya que ayuda a identificar problemas de baja mantenibilidad y facilita estrategias de refactorización, contribuyendo a la mejora general de la calidad del software.

Una dirección notable para futuras investigaciones sería evaluar empíricamente la efectividad del modelo en una gama más amplia de proyectos industriales y de código abierto. Dichas validaciones podrían proporcionar información crítica sobre cómo se pueden mejorar las funciones

de pertenencia de fuzzificación para mejorar la precisión de la evaluación. Además, las reglas de inferencia difusa utilizadas en el modelo podrían ajustarse para mejorar la evaluación de las características y subcaracterísticas de calidad. Estas mejoras permitirían que este modelo genere evaluaciones más precisas, lo que contribuiría al desarrollo de arquitectura de microservicios de mayor calidad.

Este modelo de calidad jerárquico propuesto es prometedor para evaluar la capacidad de mantenimiento de arquitectura de microservicios y guiar la investigación futura hacia la mejora de la calidad de las arquitecturas de microservicios.

Referencias

[1] S. Apel, F. Hertrampf, and S. Späthe. Towards a Metrics-Based Software Quality Rating for a Microservice Architecture: Case Study for a Measurement and Processing Infrastructure. In *Communications in Computer and Information Science*, pages 205–220. Springer Verlag, 2019.

[2] J. Bogner, S. Wagner, and A. Zimmermann. Towards a Practical Maintainability Quality Model for Service- and Microservice-based Systems. 2017.

[3] M. Cardarelli, L. Iovino, P. Di Francesco, A. Di Salle, I. Malavolta, and P. Lago. An Extensible Data-Driven Approach for Evaluating the Quality of Microservice Architectures. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, pages 1225–1234, New York, NY, USA, 2019. Association for Computing Machinery.

[4] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[5] I. O. for Standardization. *ISO/IEC 25010:2011 - Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE)*. International Organization for Standardization, Geneva, Switzerland, 2011.

[6] I. O. for Standardization. *ISO/IEC 25020:2019 - Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Quality Measurement Framework*. ISO/IEC, Geneva, Switzerland, 2019.

[7] E. Mamdani and S. Assilian. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[8] R. Yilmaz and F. Buzluca. A Fuzzy Quality Model to Measure the Maintainability of Microservice Architectures. In *2021 2nd International Informatics and Software Engineering Conference (IISEC)*, pages 1–6. IEEE, 2021.

[9] R. Yilmaz and F. Buzluca. A Fuzzy Logic-Based Quality Model for Identifying Microservices with Low Maintainability. *Journal of Systems and Software*, 2024.

[10] Y. Zhou, B. Xu, and H. Leung. On the Ability of Complexity Metrics to Predict Fault-Prone Classes in Object-Oriented Systems. *Journal of Systems and Software*, 83:660–674, May 2010.