EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación
y Desarrollo Tecnológico

Departamento de Ciencias Computacionales
Maestría en Ciencias de la Computación

TESIS DE MAESTRÍA

Modelo de calidad para determinar la granularidad de microservicios mediante el balance de atributos

presentada por

Ing. Ariel Martínez Roque

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Juan Carlos Rojas Pérez

Cuernavaca, Morelos, México. Enero de 2025.

cenidet®
Centro Nacional de Investigación y Desarrollo Tecnológico

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación
y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Mor., 03/diciembre/2024
OFICIO/DCC/312/2024

Asunto: Aceptación de documento de tesis
CENIDET-AC-M14-OFICIO

CARLOS MANUEL ASTORGA ZAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes del Comité Tutorial de **ARIEL MARTÍNEZ ROQUE**, con número de control M23CE052, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"MODELO DE CALIDAD PARA DETERMINAR LA GRANULARIDAD DE MICROSERVICIOS MEDIANTE EL BALANCE DE ATRIBUTOS"**, y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

ATENTAMENTE
Excelencia en Educación Tecnológica·
"Conocimiento y Tecnología al Servicio de México"

Dr. Juan Carlos Rojas Pérez
Director de tesis

Dra. Olivia Graciela Fragoso Díaz
Revisor 1

M.C. Humberto Hernández García
Revisor 2

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación
y Desarrollo Tecnológico
Subdirección Académica

Cuernavaca, Mor., 11/diciembre/2024
No. De Oficio: SAC/382/2024
Asunto: Autorización de impresión de tesis

ARIEL MARTÍNEZ ROQUE
CANDIDATO AL GRADO DE MAESTRO
CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"MODELO DE CALIDAD PARA DETERMINAR LA GRANULARIDAD DE MICROSERVICIOS MEDIANTE EL BALANCE DE ATRIBUTOS"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica-
"Conocimiento y Tecnología al Servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

Dedicatoria

A mis padres por el apoyo incondicional en todo momento.

A mi hermana por el amor infinito y fuerzas brindados cada día.

A mi pareja por apoyarme, ayudarme y ofrecer su amor en todo momento.

A mi hijo por ser el motor impulsor

Agradecimientos

Al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCyT) por el programa Sistema Nacional de Posgrados (SNP) por medio del cual me apoyó económicamente como estudiante de tiempo completo.

Al Tecnológico Nacional de México campus Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), por brindarme la oportunidad de pertenecer a su comunidad estudiantil.

A mi director de tesis, el Dr. Juan Carlos Rojas Pérez, por el apoyo y seguimiento constante durante el desarrollo de esta tesis. También por su rigurosidad y su compromiso con el trabajo.

A mi comité tutorial conformado por la Dra. Olivia Graciela Fragoso Díaz y el M.C Humberto Hernández García, quienes dedicaron parte de su tiempo a las revisiones de este trabajo y ayudaron a la mejora de este.

A mis padres, abuela, hermana, cuñado, esposa, suegra y al resto de mi familia, gracias por su apoyo incondicional y alentarme a seguir adelante.

A los profesores y personal del CENIDET por su apoyo en momentos difíciles.

Resumen

Una de las relativamente recientes arquitecturas utilizadas en el desarrollo de sistemas de software es la Arquitectura de Microservicios (MSA). Una MSA aboga por descomponer un monolito en un conjunto de pequeños servicios y hacer que se comuniquen entre sí a través de mecanismos ligeros. [1]. Descomponer un monolito en conjuntos pequeños de servicios es un reto que afrontan los microservicios.

En esta investigación se identifican, en la literatura, atributos de calidad que se usan para razonar sobre la granularidad en microservicios, se utilizan métricas ya definidas en trabajos relacionados para medir los atributos de calidad y se determina la granularidad de microservicios mediante el balance de atributos de calidad, en este caso, cohesión y complejidad. Además, se diseñó un modelo de calidad mediante el cual se puede determinar cómo se comportan los atributos de calidad cohesión y complejidad con respecto a la granularidad.

Para mostrar el comportamiento de los atributos de calidad cohesión y complejidad se ejecutaron 6 casos de prueba en 3 aplicaciones descargadas de GitHub. Las pruebas realizadas consistieron en calcular los valores de las métricas identificadas para los atributos granularidad, cohesión y complejidad. Este proceso se realizó dos veces para cada caso de estudio: en la arquitectura original y en la arquitectura modificada. Las modificaciones en la arquitectura se hicieron atendiendo los valores de granularidad y sus umbrales 0,2 y 0.6.

Las pruebas realizadas arrojaron que los valores de los atributos de calidad cohesión y complejidad se contraponen, o sea que tienen una relación inversamente proporcional entre ellos. Además, se pudo observar que los valores de cohesión mantienen una relación directamente proporcional con los valores del atributo de calidad granularidad. Por otro lado, los valores de complejidad se comportan de forma inversamente proporcional a los valores del atributo granularidad.

El presente trabajo muestra una guía para el desarrollo de microservicios que sean granulares atendiendo a un balance entre cohesión y complejidad. La investigación proporciona a los arquitectos, analistas y desarrolladores de software una herramienta para tomar decisiones en cuanto al tema de la granularidad óptima de los microservicios. Con el presente trabajo se podrá decidir en cómo balancear los atributos de calidad (en este caso cohesión y complejidad) para obtener una granularidad entre los valores más aceptados según la literatura relacionada con el tema.

Abstract

One of the relatively recent architectures used in software system development is the Microservices Architecture (MSA). An MSA advocates decomposing a monolith into a set of small services and having them communicate with each other through lightweight mechanisms [1]. Decomposing a monolith into small sets of services is a challenge faced by microservices.

In this research, quality attributes used to reason about granularity in microservices are identified in the literature, metrics already defined in related works are used to measure quality attributes, and the granularity of microservices is determined by balancing quality attributes, in this case, cohesion and complexity. In addition, a quality model was designed through which it can be determined how the quality attributes cohesion and complexity behave with respect to granularity.

To show the behavior of the quality attributes cohesion and complexity, 6 test cases were executed in 3 applications downloaded from GitHub. The tests performed consisted of calculating the values of the metrics identified for granularity, cohesion and complexity attributes. This process was performed twice for each case study: in the original architecture and in the modified architecture. The modifications to the architecture were made taking into account the granularity values and their thresholds 0.2 and 0.6.

The tests performed showed that the values of the quality attributes cohesion and complexity are opposed, that is, they have an inversely proportional relationship between them. In addition, it was observed that the cohesion values maintain a directly proportional relationship with the values of the quality attribute granularity. On the other hand, the complexity values behave in an inversely proportional way to the values of the attribute granularity.

This work shows a guide for the development of microservices that are granular, taking into account a balance between cohesion and complexity. The research provides architects, analysts and software developers with a tool to make decisions regarding the issue of optimal granularity of microservices. With this work, it will be possible to decide how to balance the quality attributes (in this case cohesion and complexity) to obtain a granularity between the most accepted values according to the literature related to the subject.

Contenido

Índice de tablas

Índice de figuras

1. Introducción

En esta sección se presenta una introducción al tema, así como una revisión de los antecedentes relevantes que motivaron este estudio. Se describe el problema de investigación que originó el presente trabajo. También se detallan los objetivos (general y específicos) que se definieron para orientar la investigación y se proporcionan el alcance y las limitaciones del proyecto. Asimismo, se justifica la importancia y relevancia de este estudio, subrayando las posibles contribuciones al campo de conocimiento.

El uso de los microservicios en el desarrollo de software presenta nuevos desafíos. Esto se debe al hecho de constituir una arquitectura relativamente reciente [1]. En este trabajo, el autor explica que una Arquitectura de Microservicios (AMS) aboga por descomponer un monolito en un conjunto de pequeños servicios y hacer que se comuniquen entre sí a través de mecanismos ligeros. También, el autor afirma que descomponer un monolito es uno de los desafíos que afronta las AMS.

En [2], los autores expresan que a pesar del creciente interés de las industrias de software en los microservicios; aún existe una falta general de enfoques sistemáticos que modelen las decisiones de diseño. Por otro lado los autores manifiestan la falta de consenso en la determinación de los límites óptimos de los microservicios: el nivel óptimo de granularidad de los microservicios.

Desde SOA se habla de los problemas que acarrea una granularidad inapropiada. Según los autores en [3], algunos de los problemas que pueden surgir debido a una granularidad de servicio incorrecta son los siguientes:

- ✓ Duplicación de servicios (Diferentes servicios para tareas similares)
- ✓ Dificultad en el mantenimiento.
- ✓ La gobernanza del servicio es extremadamente difícil cuando la cantidad de servicios es enorme y los servicios son simplemente interfaces de grano fino.
- ✓ La reutilización de servicios entre aplicaciones se ve afectada.

1.1. Antecedentes

Determinar la granularidad de un microservicio es un problema que puede ser abordado desde diferentes enfoques tales como el estructural y de comportamiento [4]. Varios trabajos se han

propuesto para definir la granularidad de los microservicios, así como para determinar y medir los atributos de calidad (AC); sin embargo, ninguno se ha enfocado a determinar la granularidad a partir del balance entre los AC. Determinar el balance adecuado entre los AC aplicables a microservicios en diversos escenarios puede constituir un nuevo enfoque aplicable a los diversos trabajos que abordan el tema de la granularidad.

En [1] se realizó una Revisión Sistemática de la Literatura (RSL) donde se identificaron seis AC más críticos de AMS, este trabajo presenta 19 tácticas que abordan arquitectónicamente los seis AC (usando una plantilla que incluye introducción, motivación, descripción, restricción y dependencia). Por otro lado, en [5] los autores presentan una técnica sistemática para identificar patrones arquitectónicos que afectan un **conjunto** dado de atributos de calidad, el balance se aplica a patrones. Para demostrar su uso, los autores realizan cinco pasos propuestos en el contexto del estilo arquitectónico de microservicios, identificando patrones de microservicios arquitectónicos que influyen en las decisiones de diseño como el tamaño del servicio, el uso compartido de la base de datos y el acoplamiento del servicio.

Por otro lado en [6], los autores presentan un balance de patrones realizando 9 entrevistas semiestructuradas para recopilar la experiencia de la industria con respecto a (1) el conocimiento y la adopción de patrones de software, (2) los balances arquitectónicos percibidos de los patrones y (3) las métricas que usan los profesionales para medir los atributos de calidad. Estos trabajos no presentan un balance general de los atributos necesarios en el desarrollo de microservicios, todos estos son estudios enfocados en general a problemas específicos en microservicios.

1.2. Planteamiento del problema

Establecer la granularidad en un microservicio es un reto que ha sido y sigue siendo abordado en la actualidad. Un microservicio con una inadecuada granularidad influye o afecta entre otros atributos de calidad, al rendimiento, escalabilidad, cohesión y complejidad. El problema de la determinación de la granularidad ha sido abordado desde diversos enfoques. En los trabajos propuestos no se ha abordado analizar y determinar el balance entre atributos de calidad, lo cual puede constituir una nueva alternativa de solución para abordar el problema de la granularidad en microservicios.

1.2.1. Objetivos

1.2.2. Objetivo general

Proponer un modelo de calidad para determinar la granularidad de microservicios considerando el balance entre atributos.

1.2.3. Objetivos específicos

a. Establecer cuáles atributos de calidad son afectados por la granularidad en microservicios.
b. Utilizar métricas ya establecidas en la literatura para medir atributos de calidad.
c. Determinar granularidad de microservicios mediante el balance de atributos de calidad.

1.3. Alcances y limitaciones

1.3.1. Alcances

a. El balance se realizó entre dos atributos de calidad aplicables para determinar la granularidad de los microservicios.
b. Se utilizaron métricas ya definidas en la literatura para medir atributos de calidad de software.

1.3.2. Limitaciones

a. Las pruebas de la investigación realizaron con base en repositorios de software con microservicios ya implementados.
b. Se utilizaron herramientas de código abierto.

1.2. Estructura del documento

El resto del documento se encuentra organizado de la siguiente manera, en el Capítulo 2 se presenta el marco conceptual donde se contextualiza al lector sobre los temas que se tratan en el documento, el Capítulo 3 presenta una recopilación los trabajos relacionados y su análisis, en el Capítulo 4 se presentan las etapas que conforman el proceso para el desarrollo del modelo de calidad propuesto, en Capítulo 5 se muestran los casos de estudio evaluados, además se presentan las pruebas y resultados obtenidos. Por último, se presentan las conclusiones y trabajos futuros.

2. Marco Conceptual

En este capítulo se presentan los principales conceptos y fundamentos abordados en la presente investigación.

2.1. Microservicios

Son un enfoque para desarrollar una aplicación compuesta por pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose a través de mecanismos ligeros. Estos servicios se construyen alrededor de las capacidades de negocio y se despliegan de forma independiente. Cada funcionalidad se encapsula en un servicio que puede escalar de forma diferente de acuerdo con sus necesidades, a diferencia de las aplicaciones monolíticas donde se debe replicar el monolito completo [7].

Los microservicios se presentan como un enfoque de implementación de la Arquitectura Orientada a Servicios (SOA), estos tienen como objetivo mejorar las desventajas y los problemas presentados en SOA [8].

2.2. Arquitectura basada en microservicios

Establece construir una aplicación como un conjunto de servicios, donde cada uno de estos es independiente del otro y hasta pueden ser escritos en lenguajes diferentes y mantenidos por equipos diferentes. Lo anterior permite que el problema de escalar se solucione incrementando el procesamiento necesario para el servicio específico que lo esté requiriendo y no de otros que no lo necesitan [9].

2.3. Atributos de calidad de software

Entiéndase los atributos de calidad de un sistema de software como las propiedades o cualidades de calidad que dicho sistema debe satisfacer. [10].

Según la norma ISO/IEC 25010 en [11] la calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Dichos requisitos se encuentran representados en el modelo de calidad de la norma ISO/IEC 25010.

Este modelo categoriza la calidad del producto en características y subcaracterísticas. En la Tabla 1 se detallan los requisitos del modelo de calidad de la ISO/IEC 25010.

Tabla 1. Atributos de calidad según la ISO/IEC 25010 [11].

Requisitos	Especificación
Adecuación Funcional	Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas de los usuarios cuando el producto se usa en las condiciones especificadas
Eficiencia de Desempeño	Esta característica representa el desempeño de un producto en la realización de sus funciones dentro de unos parámetros de tiempo y rendimiento especificados y con un uso eficiente de recursos (CPU, memoria, almacenamiento, energía...) utilizados bajo determinadas condiciones.
Compatibilidad	Capacidad de un producto de intercambiar información con otros productos y/o llevar a cabo sus funciones requeridas cuando comparten un mismo entorno y recursos.
Capacidad de Interacción	Capacidad del producto software para que el usuario interactúe mediante su interfaz intercambiando información para completar determinadas tareas.
Fiabilidad	Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados sin interrupciones o fallos
Seguridad	Capacidad de protección de la información y los datos de manera que las personas u otros productos tengan el grado de acceso a los datos adecuado a sus tipos y niveles de autorización, y para defenderse de los patrones de ataque de agentes malintencionados.
Flexibilidad	Capacidad del producto para adaptarse a cambios en sus requisitos, contextos de uso o entorno del sistema.
Protección	Esta característica representa la capacidad del producto, en condiciones definidas, de evitar un estado en el que se ponga en peligro la vida humana, la salud, la propiedad o el medio ambiente.
Mantenibilidad	Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectiva

En la presente investigación se hace un estudio de la relación entre los atributos de calidad cohesión y complejidad que según el autor en [12] son métricas que miden el atributo de calidad mantenibilidad de la ISO/IEC 25010.

2.4. Métrica

De acuerdo con el Institute of Electrical and Electronics Engineers (IEEE) en [13], una métrica de calidad de software es "una función cuyas entradas son datos de software y cuya salida es un solo valor numérico que puede ser interpretado como el grado en que el software posee un atributo dado que afecta su calidad".

2.5. Balance de atributos de calidad

Según la RAE[1] se define balance como un estudio comparativo de las circunstancias de una situación, o de los factores que intervienen en un proceso, para tratar de prever su evolución.

A partir de los conceptos anteriormente expuestos (balance y atributos de calidad de software) se puede definir el término balance de atributos de calidad como un estudio comparativo de propiedades o cualidades de calidad que la aplicación o software debe satisfacer para tratar de prever su comportamiento.

2.6. Modelo de Calidad

Conjunto definido de características y de relaciones entre ellas, que proporciona un marco para especificar requisitos de calidad y evaluar la calidad ISO 25010. Un modelo de calidad provee una taxonomía de factores de calidad y medidas asociadas a los mismos, apropiadas para evaluar la calidad de un sistema de software y puede ser utilizado para establecer requisitos del sistema [14].

2.7. Granularidad

La granularidad de los microservicios se define principalmente por su tamaño o dimensiones, es decir, la cantidad de operaciones expuestas por el microservicio. El objetivo es tener bajo acoplamiento, baja complejidad y alta cohesión entre los microservicios [4].

2.8. Cohesión

Los autores en [15] exponen que la cohesión se refiere al grado en el que los datos y los métodos de una clase están relacionados entre sí. En el caso de la presente investigación se basa en el grado de relación de las operaciones en un microservicio.

[1] https://www.rae.es/drae2001/balance

2.9. Complejidad

Según los autores en [15] la complejidad es el número de operaciones por microservicio. Es importante destacar que no es la única definición de complejidad. En la presente investigación el valor del atributo de calidad complejidad será el número de operaciones por microservicios.

Refiriéndose a otras definiciones de complejidad el autor en [12] dice que: “algunos ejemplos son la complejidad ciclomática de McCabe [McCabe, 1976] y las métricas de Halstead [Halstead, 1977]. Estas métricas se basan en el número de caminos de decisión independientes a tomar en un programa, y en la longitud y el vocabulario del programa, respectivamente.”

3. Trabajos Relacionados

En esta sección se presenta un análisis de investigaciones que tratan el problema de la granularidad en microservicios y su influencia sobre los atributos de calidad.

La literatura analizada se puede clasificar en 4 principales grupos: análisis de la granularidad en tiempo de diseño, en código estático, en tiempo de ejecución y mapeos sistemáticos. El análisis está orientado a encontrar atributos de calidad que son afectados por la granularidad, así como las métricas para medirlos y sus correspondientes umbrales.

3.1. Trabajos relacionados con análisis de la granularidad en tiempo de ejecución

En [2] se presenta Ambient-PRISMA Textual Language como un potencial lenguaje para definición de microservicios, concebido para realizar el cambio dinámico de la granularidad de los microservicios en tiempo de ejecución. En este trabajo no se hace un análisis de la calidad de las divisiones hechas ni tienen en cuenta atributos de calidad.

En [16], se hace mención del potencial de la granularidad para influir en la latencia de las aplicaciones. Los autores comparan el comportamiento del rendimiento de una aplicación basada en microservicios en cuatro escenarios posibles utilizando la herramienta Dynatrace (https://www.dynatrace.com). Las pruebas en esta investigación se realizaron en tiempo de ejecución.

En [17], los autores determinan la granularidad en tiempo de ejecución haciendo un balance entre las características costo del aseguramiento de la calidad y costo de despliegue. Tampoco aquí se analizan atributos de calidad ni métricas relacionadas. Además, es un trabajo basado en prueba y error de experiencias técnicas realizadas a una aplicación para descomponerla en microservicios.

Por otro lado, en [18] se hace un análisis de granularidad para Function Block (Bloque de Funciones) como servicios. En este trabajo no se especifican métricas para medir granularidad en microservicios.

Más detalles de los trabajos abordados en esta sección se pueden encontrar en Anexo 1.

3.2. Trabajos relacionados con análisis de la granularidad en tiempo de diseño

En [8] el autor propone un modelo matemático para determinar la granularidad en microservicios en tiempo de diseño, basándose en historias de usuario. Sin embargo en el trabajo no se especifica umbrales para las métricas que utilizan en el modelo.

Asimismo en [19] el autor propone una metodología para determinar la granularidad en tiempo de diseño a partir de la especificación de casos de usos de la aplicación. En este trabajo solo se analiza el diseño de la aplicación por lo que no se hace un análisis de métricas basado en código fuente.

Más detalles de los trabajos abordados en esta sección se pueden encontrar en el Anexo 2.

3.3. Mapeos sistemáticos de la literatura

Los autores en [4] hacen un mapeo sistemático de la literatura evaluando la transición de un monolito a una arquitectura de microservicios, aquí el autor define el término microservilización refiriéndose a la antes mencionada transición. Se plantea que uno de los problemas fundamentales de la transición a los microservicios es analizar su nivel de granularidad. En este trabajo también se hace un análisis de los atributos de calidad más recurrentes en la literatura analizada para razonar sobre la granularidad en microservicios (escalabilidad, rendimiento, mantenibilidad y fiabilidad). En este trabajo no se especifican métricas para medir los atributos de calidad expuestos.

Más detalles de los trabajos abordados en esta sección se pueden encontrar en el Anexo 3.

3.4. Trabajos relacionados con el análisis de la granularidad en código estático.

Por otro lado, en [20] los autores utilizan una versión modificada de la métrica de falta de cohesión de Henderson-Sellers, para que sea adecuada para el diseño de microservicios. En este caso las métricas son adaptadas para medir atributos de calidad como cohesión y complejidad. Se destaca de este artículo que se hace una medición de los atributos de calidad para valorar la posible descomposición de un microservicio en dependencia de los valores arrojados en el análisis de las métricas descritas en él. Las pruebas se realizan en repositorios de aplicaciones basadas en microservicios ya implementadas.

A propósito, en [15] los autores presentan un marco de métricas para evaluar la calidad de los diseños de arquitectura de microservicios. Las métricas propuestas incluyen la Métrica de Granularidad de Servicio (SGM), Métrica de Falta de Cohesión (LCOM) y Número de Operaciones (NOO), que miden

la granularidad, cohesión y complejidad de los microservicios individuales mediante el análisis de la interfaz de programación de aplicaciones (API). En este trabajo se proponen umbrales o valores óptimos para cada métrica aplicada.

Algo similar hace el autor en [12] con la diferencia que se evalúa la granularidad desde el punto de vista del atributo de calidad mantenibilidad. En el trabajo se hace un análisis de código estático teniendo en cuenta las diferentes versiones de una aplicación. En este último se utilizan métricas y umbrales descritos en [21].

En [21] el autor utiliza el enfoque RAMA (RESTful API Metric Analyzer) para calcular las métricas de mantenibilidad utilizando APIs RESTful. El enfoque propone un análisis estático de las descripciones de API RESTful para calcular métricas estructurales relacionadas con la mantenibilidad del servicio. Continuando con [21], los autores determinan métricas de mantenibilidad relacionadas con la complejidad, la cohesión y el tamaño del servicio. Además, se diseñó un estudio de umbrales basado en cuartiles. Este estudio se basó en la recopilación de métricas de 1,737 APIs RESTful públicas para derivar umbrales de métricas.

3.5. Tabla comparativa de trabajos relacionados

En la Tabla se relacionan temas que presentan los trabajos seleccionados y que, a su vez, son afines con el presente estudio.

Los encabezados de la Tabla se definen como: **AMS:** Arquitectura de Microservicios, **MC:** Modelo de Calidad, **Grd:** Granularidad, **AC:** Atributos de calidad **B-AC:** Balance de Atributos de Calidad.

Tabla 2. Trabajos relacionados

No.	***Título***	***AMS***	***MC***	***Grd***	***AC***	***B-AC***	***Solución***
1	*A Method for Architectural Trade-off Analysis Based on Patterns: Evaluating Microservices Structural Attributes* [5]	Sí	No	No	Sí	Sí	Metodología
2	*Microservice Ambients:An Architectural Meta-modelling Approach for Microservice Granularity* [2]	Sí	No	Sí	No	No	Enfoque para calcular granularidad
3	*From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture*[17]	Sí	No	Sí	Sí	Sí	Lección técnica aprendida
4	*Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios.* [8]	Sí	No	Sí	Sí	No	Modelo
5	*Metodología en la determinación de la granularidad de un Microservicio* [19]	Sí	No	Sí	No	No	Metodología
6	*Microservice Transition and its Granularity Problem: A Systematic Mapping Study.* [4]	Sí	No	Sí	Sí	No	Estudio de mapeo sistemático
7	*Granularity Cost Analysis for Function Block as a Service* [18]	Sí	No	Sí	Sí	No	Análisis
8	*Workload-based Clustering of Coherent Feature Sets in Microservice Architectures*[22]	Sí	No	Sí	Sí	Sí	Enfoque
9	*The Role of Service Granularity in a Successful SOA Realization A Case Study*[3]	No	No	Sí	No	No	Estudio de caso
10	*Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications*[23]	Sí	No	Sí	Sí	No	Set de atributos de calidad
11	*Microservices: Granularity vs. Performance*[16]	Sí	No	Sí	No	No	Pautas
12	*Selección y adaptación de métricas para Microservicios*[24]	Sí	No	Sí	Sí	No	Adaptación.
13	*MicroValid: A Validation Framework for Automatically Decomposed Microservices*[25]	Sí	No	Sí	Sí	No	Marco de código abierto

No.	***Título***	***AMS***	***MC***	***Grd***	***AC***	***B-AC***	***Solución***
14	*A Systematic Evaluation of Microservice Architectures Resulting from Domain-Driven and Dataflow-Driven Decomposition*[26]	Sí	No	Sí	Sí	No	Evaluación
15	*Microservices: The Evolution and Extinction of Web Services?*[27]	Sí	No	Sí	No	No	Revisión sistemática
16	*Size Matters: Microservices Research and Applications*[28]	Sí	No	Sí	No	No	Revisión sistemática
17	*Defining and measuring microservice granularity*[29]	Sí	No	Sí	Sí	No	Revisión sistemática
18	*Extracting Microservices' Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach*[20]	Sí	No	Sí	Sí	No	Método de descomposición y evaluación de aplicaciones basadas en microservicios
19	*A case study on measuring the size of microservices*[30]	Sí	No	Sí	Sí	No	Estudio de caso
20	*A quantitative assessment method for microservices granularity to improve*[12]	Sí	No	Sí	Sí	No	Enfoque para evaluación cuantitativa de granularidad
21	*A Metrics Framework for Evaluating Microservices Architecture Designs*[15]	Sí	No	Sí	Sí	No	Set de métricas
22	*Collecting Service-Based Maintainability Metrics from RESTful API Descriptions: Static Analysis and Threshold Derivation*[21]	Sí	No	No	Sí	No	Enfoque para calcular métricas
	Modelo de calidad para determinar la granularidad de microservicios mediante el balance de atributos.	**Sí**	**Sí**	**Sí**	**Sí**	**Sí**	**Modelo de Calidad**

A partir del análisis realizado en los trabajos relacionados se concluye que en la mayoría de estos no se detallan las métricas utilizadas [2], [16], [17], [18]. Por otro lado, en los trabajos donde los autores sí miden granularidad en tiempo de diseño [19], no se hace referencia a atributos de calidad ni a métricas relacionadas. En [8] el autor propone varias métricas basado en la literatura y otras adaptadas para el uso en microservicios, sin embargo, no define umbrales para dichas métricas. En el mapeo sistemático de la literatura consultado tampoco se mencionan o describen umbrales para las métricas expuestas. En relación a los trabajos en los que se mide granularidad en código estático [12], [15], [20], [21] sí se definen umbrales para las métricas utilizadas sobre todo basados en la literatura o

haciendo adaptaciones de métricas utilizadas en la POO, sin embargo, carecen de un modelo de calidad donde se relacionen todos los elementos para determinar la granularidad en microservicios mediante un balance de atributos de calidad.

3.6. Trabajos adicionales

Para una mejor comprensión del tema abordado en la presente investigación se consultaron los trabajos [7], [9], [21], [21, p. 20], [21], [28], [31], [32],[5], [33], [34], [35], [36], [37], [38], [39], [40], [13], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50].

Estos trabajos fueron consultados para temas teóricos sobre microservicios, calidad de software y métricas.

4. Desarrollo de la solución

En esta sección se presenta el proceso de elaboración del modelo de calidad. Para ello se analizan los atributos de calidad que más se repiten en la literatura cuando se razona sobre la granularidad en microservicios. También se seleccionarán las métricas y umbrales definidos en la literatura.

4.1. Modelo de calidad

Para el proceso de elaboración del modelo de calidad se definen los atributos de calidad que forman parte del modelo a partir de un análisis de la literatura consultada. Por otro lado, se identifican las métricas que permiten medir cada uno de los atributos identificados. Además, se obtiene los umbrales definidos para cada una de las métricas identificadas.

4.1.1. Criterios para la selección de métricas y atributos de calidad

1. Cantidad de trabajos que mencionan la relación del atributo de calidad con la granularidad.
2. Las métricas deberán ser medibles en código fuente estático sin necesidad de la documentación del proyecto.
3. 1234.

En la Figura 1 se muestra el proceso de elaboración del modelo de calidad.

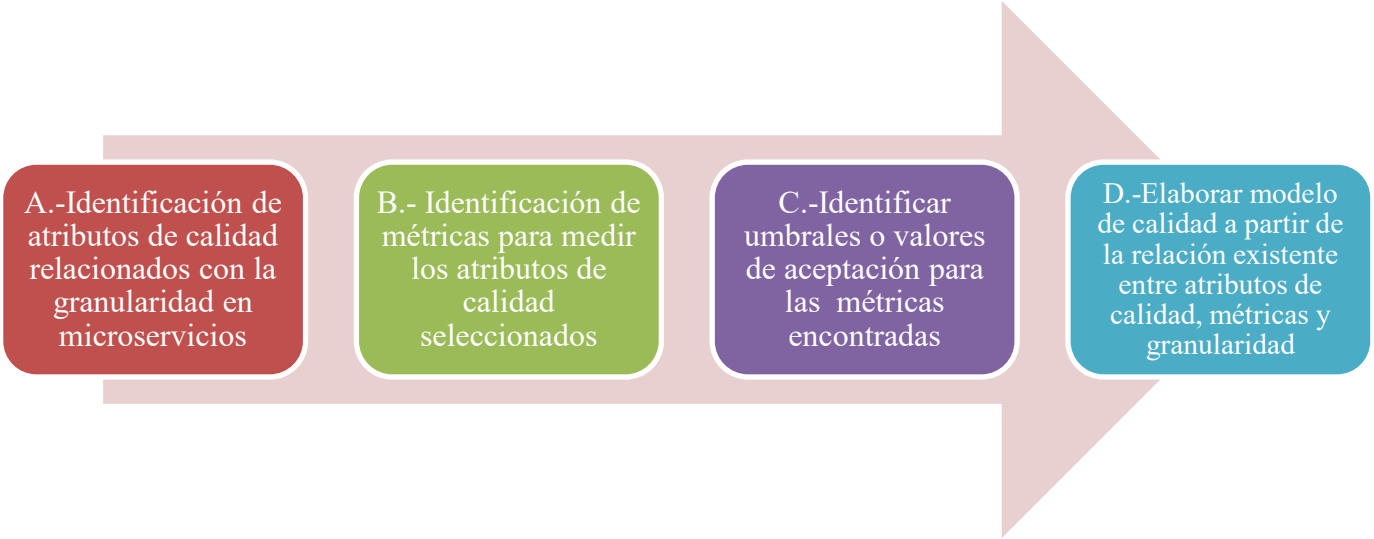

Figura 1. Proceso de elaboración del modelo de calidad

4.2. A.- Atributos de calidad relacionados con la granularidad en microservicios

A partir del análisis de trabajos relacionados en la Tabla se seleccionaron los trabajos que tratan atributos de calidad y su relación con la granularidad en microservicios. Estos trabajos y los atributos de calidad se presentan en la Tabla 3.

Tabla 3. Trabajos que relacionan atributos de calidad con la granularidad en microservicios

Referencia	***Título***	***Atributos de calidad***
[8]	Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios.	Modularidad, Funcionalidad, Mantenibilidad, Disponibilidad, Seguridad, Fiabilidad, Rendimiento, Escalabilidad, Acoplamiento
[4]	Microservice Transition and its Granularity Problem: A Systematic Mapping Study	Rendimiento, Escalabilidad, Mantenibilidad, Complejidad
[18]	Granularity Cost Analysis for Function Block as a Service	Complejidad
[22]	Workload-based Clustering of Coherent Feature Sets in Microservice Architectures	Rendimiento, Escalabilidad
[23]	Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications	Acoplamiento, Cohesión
[20]	Extracting Microservices' Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach	Cohesión, Complejidad
[15]	A Metrics Framework for Evaluating Microservices	Cohesión, Complejidad

Tabla 3. Trabajos que relacionan atributos de calidad con la granularidad en microservicios. (Continuación)

Referencia	***Título***	***Atributos de calidad***
[12]	A quantitative assessment method for microservices granularity to improve maintainability	Mantenibilidad, Complejidad, Cohesión, Acoplamiento
[21]	*Collecting Service-Based Maintainability Metrics from RESTful API Descriptions: Static Analysis and Threshold Derivation*.	Mantenibilidad, Complejidad, Cohesión

En la Figura 2 se muestra una taxonomía de atributos de calidad asociados a la granularidad en microservicios.

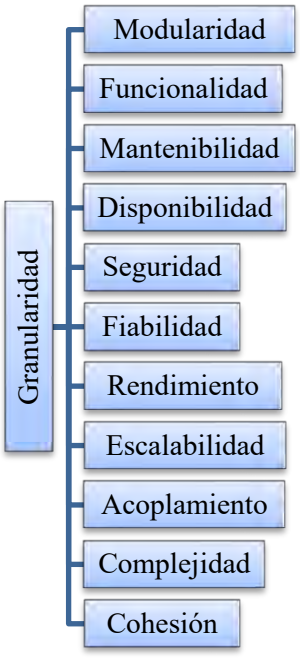

Figura 2. Taxonomía de atributos de calidad relacionados con la granularidad en microservicios

El gráfico de la Figura 3 representa la frecuencia con que aparecen los atributos de calidad en los trabajos relacionados analizados.

La frecuencia con la que aparecen los atributos de calidad en los trabajos relacionados es:

- ✓ Complejidad (en 6 de los trabajos analizados)
- ✓ Cohesión (en 5 de los trabajos analizados)
- ✓ Mantenibilidad (en 5 de los trabajos analizados)
- ✓ Escalabilidad (en 3 de los trabajos analizados)
- ✓ Rendimiento (en 3 de los trabajos analizados)
- ✓ Acoplamiento, fiabilidad, seguridad, disponibilidad funcionalidad, modularidad (en 1 de los trabajos analizados)

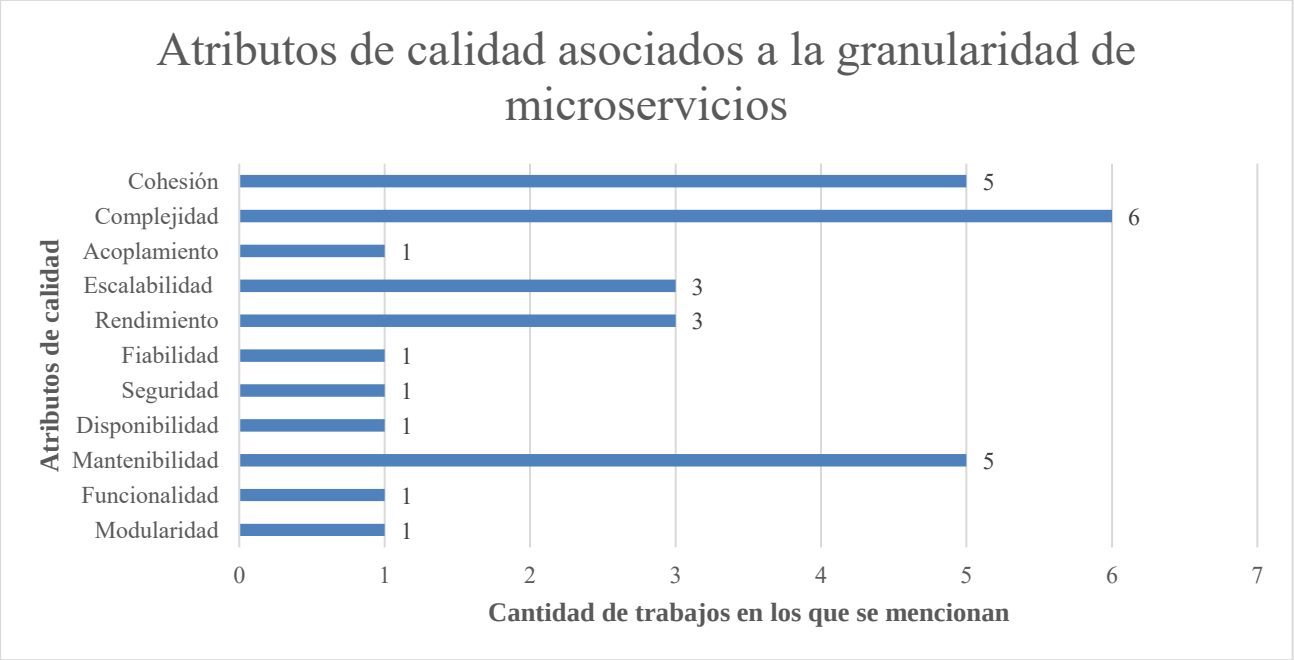

Figura 3. Trabajos que asocian atributos de calidad a la granularidad en microservicios

4.3. B.- Métricas para medir los atributos de calidad seleccionados

Se efectuó una búsqueda en la literatura de métricas para calcular los atributos de calidad que se muestran en la Figura 2 y que se relacionan con la granularidad en microservicios (complejidad, cohesión, mantenibilidad, escalabilidad, rendimiento, acoplamiento, fiabilidad, seguridad, disponibilidad funcionalidad, modularidad). En la búsqueda realizada se encontraron métricas para medir algunos de los atributos de calidad (cohesión, rendimiento, complejidad, granularidad y acoplamiento) mostrados en la taxonomía de la Figura 2.

4.3.1. Métricas para cohesión.

Para el caso del atributo de calidad Cohesión se encontraron las siguientes métricas:

- ✓ **LCOM** (Falta de cohesión).
- ✓ **SIDC** (Cohesión de datos de Interface de Servicio)

En la presente investigación se utiliza la métrica LCOM siguiendo sus valores de aceptación o umbrales definidos en la literatura.

No se hará uso de la métrica SIDC debido a la falta de claridad de los autores en la explicación de cada uno de los parámetros utilizados en la ecuación para determinar dicha métrica.

Más detalles de las métricas de cohesión abordadas en esta sección se pueden encontrar en el Anexo 4.

4.3.2. Métricas para rendimiento

Para el caso del atributo de calidad rendimiento se encontró la métrica:

- ✓ **Avg. Calls** (Promedio invocaciones de un microservicio a otros microservicios en una aplicación)

  En el caso de la métrica Avg. Calls no aparecen umbrales definidos. Además, la métrica se mide en tiempo de ejecución. Por esta razón el atributo de calidad rendimiento queda fuera de los atributos a evaluar. Este no cumple con el criterio de selección número 2. El criterio expresa que las métricas deberán ser medibles en código fuente estático.

4.3.3. Métricas para complejidad

Para el caso del atributo de calidad complejidad se encontró la métrica:

- ✓ **NOO** (Número de operaciones) propuesta en [15] y los autores definen hasta 10 el número de operaciones como umbral de esta métrica.

El número de operaciones relacionadas con un microservicio (**NOO**) es similar a la métrica WMC, que considera el número de métodos miembro relacionados con una clase específica como una métrica de complejidad, un mayor número de métodos conduce a una mayor complejidad. En el caso de los

microservicios, el número de operaciones relacionadas con un microservicio específico es el indicador de complejidad para la aplicación de microservicios [15].

La métrica NOO es seleccionable en el contexto del presente trabajo. Los valores de esta métrica pueden ser obtenidos desde código fuente estático. También sus umbrales están definidos en la literatura. Según lo anterior cumple con los 3 criterios de selección de la sección 4.1.1.

4.3.4. Métricas para granularidad

Para el caso de la granularidad se definen métricas para calcular directamente su valor

- ✓ En [15] se propone la métrica **SGM** (Métrica de Granularidad en Servicios) para calcular directamente la granularidad como atributo de calidad. Esta se calcula a través de la sumatoria del producto de la Granularidad de Datos (**DGS**) y la Granularidad Funcional (**FGS**) atendiendo a los resultados obtenidos en diferentes aplicaciones basadas en microservicios, los autores definieron como umbral o valores de aceptación para SGM valores entre 0.2 y 0.6.

La métrica SGM es medible en código fuente estático sin necesidad de la documentación del proyecto. También tiene valores de aceptación o umbrales propuestos en la literatura para determinar los mejores valores de SGM. Esto quiere decir que está de acuerdo con los criterios de selección definidos en la sección 4.1.1.

4.3.5. Métricas para acoplamiento

Para el caso del atributo de calidad acoplamiento se encontraron las métricas:

- ✓ ***AIS*** **(Importancia absoluta del microservicio)** es el número de otros microservicios que invocan al menos una operación de la interfaz de un microservicio [8].
- ✓ ***ADS*** **(Dependencia absoluta del microservicio)** es el número de otros microservicios de los que depende el microservicio [8].
- ✓ ***SIY*** (Interdependencia de los microservicios) Número de pares de microservicios interdependientes [8].

Para el caso de las métricas antes descritas, el autor no muestra ninguna definición de umbrales o valores de aceptación y este es uno de los criterios de selección de atributos de calidad que se deben cumplir para tenerse en cuenta en el balance de atributos.

Más detalles de estas métricas de acoplamiento en el Anexo .

4.4. C. Relación entre atributos de calidad, métricas y granularidad

Es importante destacar que, aquellos atributos de calidad de los que no se hallaron métricas relacionadas no se tendrán en cuenta para el balance. Esto responde a los criterios de selección definidos en la sección 4.1.1. Teniendo en cuenta lo descrito en la sección anterior 4.3 se elabora la Tabla 4 que relaciona los atributos de calidad con sus métricas, ecuaciones para los cálculos y umbrales definidos en la literatura.

Tabla 4. Métricas para medir atributos de calidad, sus fórmulas y valores de aceptación

Atributo de Calidad	***Métrica***	***Fórmula***	***Valores de aceptación***
Acoplamiento	*AIS* [8]	$AisT = \lvert\overrightarrow{AIS}\rvert = \sqrt[2]{AIS_1^2 + AIS_2^2 + \cdots + AIS_n^2}$ **AIS:** Importancia absoluta del microservicio. **AisT:** Importancia absoluta del microservicio a nivel de aplicación **n:** Cantidad de microservicios.	Sin definir
	ADS [8]	$AdsT = \lvert\overrightarrow{ADS}\rvert = \sqrt[2]{ADS_1^2 + ADS_2^2 + \cdots + ADS_n^2}$ **ADS:** Dependencia absoluta del microservicio **AdsT:** Dependencia absoluta del microservicio a nivel de aplicación **n:** Cantidad de microservicios.	Sin definir
	SIY [8]	$SiyT = \lvert\overrightarrow{SIY}\rvert = \sqrt[2]{SIY_1^2 + SIY_2^2 + \cdots + SIY_n^2}$ **SIY:** Interdependencia de los microservicios **SiyT:** Interdependencia de los microservicios a nivel de aplicación **n:** Cantidad de microservicios	Sin definir
	CpT [8]	$CpT = \lvert\overrightarrow{Cp}\rvert = \sqrt[2]{AisT^2 + AdsT^2 + SiyT^2}$ **Cp:** Acoplamiento **CpT:** Acoplamiento a nivel de aplicación **AisT:** Importancia absoluta del microservicio a nivel de aplicación **AdsT:** Dependencia absoluta del microservicio a nivel de aplicación **SiyT:** Interdependencia de los microservicios a nivel de aplicación	Sin definir
Cohesión	*LCOM [20] [51]*	$LCOM = 1 - \frac{TP}{TO * NUP}$ **LCOM:** Falta de Cohesión **TP:** Total de parámetros **TO:** Total de operaciones **NUP:** Número único de parámetros	Entre 0 y 0.8 [51]

Tabla 4. Métricas para medir atributos de calidad, sus fórmulas y valores de aceptación. (Continuación)

Atributo de Calidad	***Métrica***	***Fórmula***	***Valores de aceptación***
Granularidad	*ALCOM* [51]	$$ALCOM = \frac{\sum_{i=1}^{n} LCOM}{NS}$$ **ALCOM:** Promedio de falta de Cohesión **LCOM:** Falta de cohesión **NS:** Número de microservicios en la aplicación **n:** Número de microservicios	Entre 0 y 0.8 [51]
	Coh [8]	$$Coh_i = \frac{LC_i}{n}$$ **Coh:** Cohesión **LC**: número de pares de microservicios que no tienen ninguna interdependencia. **n:** Número de microservicios.	Sin definir
	SIDC(S) [12]	$$\boldsymbol{SIDC(S) = \frac{OC(IS)}{OD(IS)}}$$ **SIDC:** Cohesión de datos de Interface de Servicio **OC(IS):** Número de operaciones de que tienen al menos un tipo de datos de parámetro en común. **OD:** Número total de operaciones discretas definidas. **IS:** Interface de servicio. **S:** Servicio	Entre 0 – 0.55 [51]
	WSIC [8] [52] [12]	$$WsicT = Max(WSIC_1, WSIC_2, \dots, WSIC_n)$$ **WSIC:** número de operaciones de interfaz expuestas del microservicio	Entre 5-8[52]
	DGS [51]	$$DGS = \frac{IPR}{\sum_{i=1}^{n} FP} + \frac{OPR}{\sum_{i=1}^{n} CP}$$ **DGS:** Granularidad de Datos de un Servicio. **IPR:** Cantidad de parámetros de entrada en una operación. **FP:** Total de parámetros de entrada en un microservicio. **OPR:** Cantidad de parámetros de salida en una operación. **CP:** Total de parámetros de salida en un microservicio.	Sin definir
	FGS [51]	$$FGS = \frac{OT}{\sum_{i=1}^{n} O}$$ **FGS:** Granularidad Funcional de un Servicio. **OT:** Peso para una operación específica. **O:** Suma de todos los pesos de las operaciones en un microservicio.	Sin definir
	SGM [51]	$$SGM = \sum_{i=1}^{n} DGS * FGS$$ **SGM:** Métrica Granularidad de un Servicio. **FGS:** Granularidad Funcional de un Servicio. **DGS:** Granularidad de Datos de un Servicio.	Entre 0.2 y 0.6 [51]

Tabla 4. Métricas para medir atributos de calidad, sus fórmulas y valores de aceptación. (Continuación)

Atributo de Calidad	***Métrica***	***Fórmula***	***Valores de aceptación***
Rendimiento	*Avg.Calls* [8]	$Avg.Calls = \frac{1}{n}\sum_{i=1}^{n} Calls_i$ **Calls:** número de invocaciones de ms_i a otros microservicios **n:** número de microservicios	Sin definir
Complejidad	*NOO* [20] [51]	$NOO = \sum_{i=1}^{n} M$ **NOO:** Número de Operaciones	Entre 1 y 10 [51]

Luego de analizar las métricas encontradas y sus umbrales, se define el modelo de calidad. Los atributos seleccionados son granularidad, acoplamiento, rendimiento, complejidad y cohesión. Para el caso de las métricas se considerarán las que están especificadas en la Tabla 5 (SGM, DGS, FGS, IPR, FP, OPR, CP, OT, O, LCOM, TP, TO, NUP y NOO). Para el balance de atributos de calidad se tendrán en cuenta los atributos de calidad cohesión y complejidad.

Tabla 5. Abreviaturas de métricas

Abreviatura de métrica	**Significado**
SGM	*Métrica granularidad de un servicio*
DGS	*Granularidad de datos de un servicio.*
IPR	*Cantidad de parámetros de entrada en una operación.*
FP	*Total de parámetros de entrada en un microservicio.*
OPR	*Cantidad de parámetros de salida en una operación*
CP	*Total de parámetros de salida en un microservicio*
FGS	*Granularidad funcional de un servicio*
OT	*Peso para una operación específica*
O	*Suma de todos los pesos de las operaciones en un microservicio*
WSIC	*Recuento ponderado de la interfaz de servicios*
AIS	*Importancia absoluta del microservicio*
ADS	*Dependencia absoluta del microservicio*
SIY	*Interdependencia de los microservicios*
LCOM	*Falta de cohesión*
TP	*Total de parámetros de un microservicio*
TO/NOO	*Total de operaciones de un microservicio(TO y NOO se usan de manera indistinta)*
NUP	*Número único de parámetros en un microservicio*
COH	*Grado de cohesión*
SIDC	*Cohesión de datos de interface de servicio*
AVG.CALLS	*Promedio de llamadas entre los microservicios*

4.5. D. Modelo de calidad

Con los elementos expuestos en las secciones anteriores se elaboró el modelo de calidad que se presenta en la Figura 4.

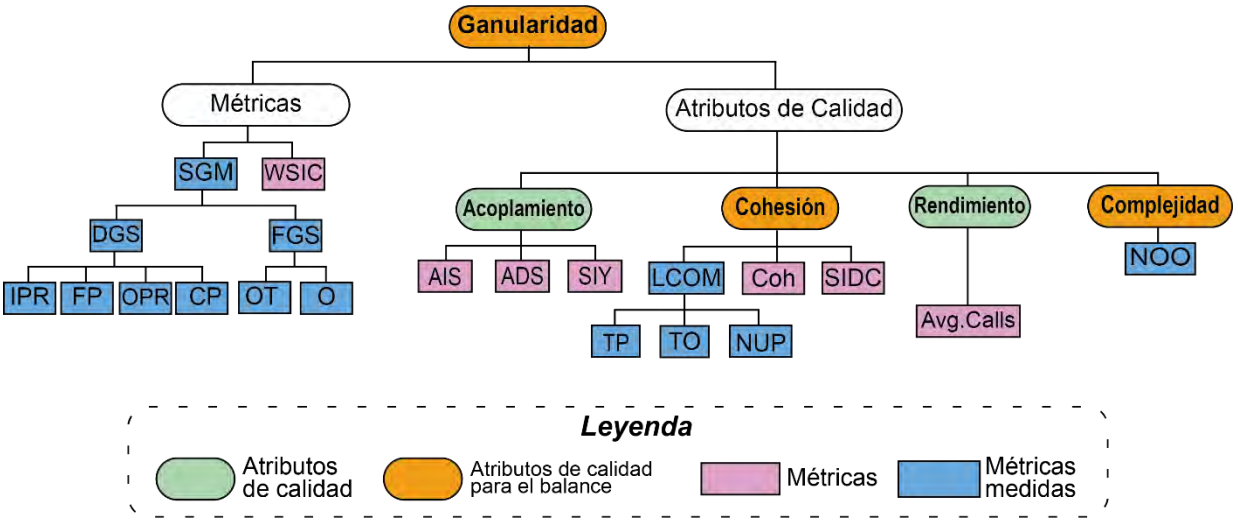

Figura 4. Modelo de Calidad

4.5.1. Especificación del modelo de calidad

En la parte superior del modelo se muestra el atributo de calidad granularidad. A la izquierda se presenta la métrica SGM (Métrica de Granularidad en Servicios) propuesta por los autores en [51] para calcular granularidad en un microservicio. Para el cálculo de SGM se necesitan los valores de DGS (Granularidad de Datos en Servicios) y FGS (Granularidad Funcional en Servicios).

Para el caso de DGS se calcula a partir de los valores de IPR (Cantidad de Parámetros de Entrada por Operación), FP (Cantidad de Parámetros de Entrada en el Microservicio), OPR (Cantidad de Parámetros de Salida por Operación). Por otro lado, FGS se determina a partir de los valores de CP (Cantidad de Parámetros de Salida en el Microservicio). Por otro lado, FGS se calcula a partir de OT (Peso de la operación específica) y O (Suma de los pesos de todas las operaciones del microservicio).

En la parte derecha se muestran atributos de calidad que se ven afectados por la granularidad en microservicios [8], [20], [51] así como las métricas utilizadas por los diferentes autores para determinar sus valores.

El modelo tiene como objetivo calcular las métricas propuestas en los trabajos consultados. También, permite determinar la granularidad adecuada de microservicios atendiendo de su relación con atributos de calidad cohesión, y complejidad. Según la bibliografía consultada [5], [8], [16], [20], [23], [30], [51], [52], la granularidad tiene influencia en los atributos antes mencionados. Por otro lado, se determinó cómo influye la granularidad en cada atributo de calidad a través de un balance de atributos de calidad, en este caso se tuvieron en cuenta los atributos de calidad marcados en naranja en la Figura 4 (cohesión y complejidad). Para la presente investigación solo se tienen en cuenta las métricas (SGM, FGS, DGS, IPR, CP, OPR, FP, OT, O, LCOM y NOO) marcadas en color azul en la Figura 4. Esto atendiendo a la identificación de sus umbrales y ecuaciones (Tabla 4) en la literatura consultada, tal y como se muestra en los criterios de selección de la sección 4.1.1.

En la Tabla 5 se muestra el significado de las abreviaturas de cada una de las métricas presentadas en el modelo de calidad propuesto en la Figura 4.

5. Pruebas y resultados

En este capítulo se presentan los casos de estudio seleccionados para probar el modelo de calidad. También se describen los casos de pruebas para cada caso de estudio. El objetivo de las pruebas es establecer el modelo de calidad mediante el balance de los atributos de calidad cohesión y complejidad.

5.1. Pruebas

En esta sección se describen las pruebas realizadas para probar el modelo propuesto a partir del balance de los atributos de calidad cohesión y complejidad. Para ello se definió un plan de prueba y los casos de pruebas.

Con las pruebas realizadas también se extraen los valores de las métricas propuestas de los repositorios de microservicios obtenidos. A partir de estas métricas se calcularon los valores siguiendo las ecuaciones propuestas en la literatura.

Una vez obtenidos estos valores se efectuaron cambios (dividir o acoplar microservicios) en la arquitectura original de las aplicaciones basadas en microservicios que se encuentran en los repositorios. Se repitió el proceso para ver cómo se comportan los valores de la granularidad (SGM) con respecto a los atributos de calidad cohesión y complejidad permitiendo, de esta forma, un balance entre estos dos atributos.

5.1.1. Elementos del modelo de calidad a ser evaluados en las pruebas

Para probar el modelo de calidad se tomaron en cuenta las métricas SGM (Métrica de granularidad en Servicio), LCOM (Falta de cohesión) y NOO (Número de operaciones) que tienen umbrales definidos de 0.2-0.6, de 0-0.2 y 1-10 respectivamente. En la Tabla 6 se listan dichas métricas y sus umbrales.

Tabla 6. Métricas para el balance de la granularidad en microservicios

Métrica	**Descripción**	**Fórmula**	**Atributo**	**Umbrales**
SGM	La Métrica de Granularidad de Servicio (SGM) se calcula a través de la sumatoria del producto de la Granularidad de Datos (DGS) y la Granularidad Funcional (FGS) donde *n* representa el número de operaciones de un microservicio.	$SGM = \sum_{i=1}^{n} DGS * FGS$ (1) $DGS = \frac{IPR}{\sum_{i=1}^{n} FP} + \frac{OPR}{\sum_{i=1}^{n} CP}$ (2) $FGS = \frac{OT}{\sum_{i=1}^{n} O}$ (3)	Granularidad	Poco Granular: **0-0.2** Valores aceptados: **0.21-0.6** Muy Granular: **0.61-1**
LCOM	La métrica de falta de cohesión mide la cohesión o, en otras palabras, la similitud entre las operaciones de un servicio específico y si estas operaciones están relacionadas entre sí. Esta se calcula dividiendo la Cantidad de Parámetros Total (TP) en un microservicio entre el producto del Número Total de Operaciones (TO) por la Cantidad de Parámetros Únicos (NUP).	$LCOM = 1 - \frac{\sum_{i=1}^{n} TP}{TO * NUP}$ (4)	Cohesión	Poco Cohesivo: **0-0.2** Valores aceptados **0.21-1**
NOO	Número de operaciones por microservicio [15], [20]	$NOO = \sum_{i=1}^{n} M$ (5)	Complejidad	Valores aceptados **1-9** Muy complejo **>=10**

5.1.2. Casos de estudio

Para la realización de las pruebas se descargaron tres aplicaciones basadas en arquitecturas de microservicios. Estos repositorios se obtuvieron de Github y están desarrollados en Java utilizando el Framework Spring Boot. En Tabla 6 se relacionan las aplicaciones seleccionadas para la realización de las pruebas en el presente trabajo .

Tabla 7. Casos de estudio

Aplicación	**Descripción**
Train Ticket : A Benchmark Microservice System [53]	Este proyecto es un sistema de reserva de boletos de tren basado en una arquitectura de microservicios que contiene 36 microservicios.
Customers, accounts and money transfers [54]	Aplicación bancaria sencilla. La arquitectura de la aplicación se basa en microservicios, Event Sourcing y CQRS. Está construida utilizando Spring Cloud, Spring Boot y la plataforma Eventuate. La aplicación se utiliza en laboratorios prácticos que forman parte de una clase de servicios de microservicios impartida por Chris Richardson.
REST Microservices architecture for E-commerce [55]	Implementación de una aplicación basada en Microservicios REST en un E-Commerce con Spring Boot, Cloud y múltiples módulos.

5.1.3. Criterios de aceptación y suspensión

Criterios de aceptación

Los criterios de aceptación se basaron la obtención de los valores de las métricas en el rango de 0 a 1 exceptuando el caso de la métrica NOO cuyo valor no tiene restricción. Los valores de las métricas se obtuvieron de manera semiautomática con el uso de una herramienta desarrollada en el transcurso de la presente investigación y Microsoft Excel.

Criterios de suspensión

Los criterios de suspensión se basaron en:

1- Obtención fuera de los valores del rango de 0 a 1.

2- Errores en la obtención de las métricas debido a código no contemplado en las gramáticas definidas en la herramienta desarrollada.

5.1.4. Entregables

Como resultado del diseño y ejecución de las pruebas se obtuvieron los siguientes documentos:

- ✓ Plan de pruebas.
- ✓ Bitácora de pruebas.

5.1.5. Liberación de las pruebas

Las pruebas se liberaron una vez se concluyeron los casos de pruebas con la documentación correspondiente para cada caso.

5.1.6. Ambiente de trabajo para la aplicación de las pruebas

Hardware

- ✓ Procesador Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
- ✓ RAM instalada 32.0 GB (31.8 GB utilizable)

Software

- ✓ Windows 11 Pro, versión 22H2
- ✓ IntellJ IDEA 2023.2.5
- ✓ Docker versión 24.0.6, build ed223bc
- ✓ Docker Desktop versión 4.24.2
- ✓ JDK 19
- ✓ Navegador web
- ✓ Microsoft Excel

5.1.7. Diseño de pruebas

Las pruebas permiten mostrar la relación de la granularidad con los atributos de calidad cohesión y complejidad. Por otro lado, se determinan los valores de las métricas propuestas en aplicaciones basadas en una arquitectura de microservicios. Por último, se hace un balance de los atributos de calidad cohesión y complejidad. Al final de la aplicación del modelo de calidad los valores de los atributos de calidad deben estar en el rango de valores de aceptación o umbrales (LCOM: 0-0.2, NOO: 1-9, SGM: 0.2-0.6) definidos en la Tabla 6.

5.1.7.1. Escenarios

Las pruebas se aplicaron en dos de cuatro escenarios dependiendo del valor arrojado de la métrica SGM descrita en el modelo de calidad (Figura 4). En cada uno de los escenarios se determinaron, además, los valores de las métricas NOO y LCOM (Figura 5).

1- Determinar el valor de las métricas (SGM, LCOM y NOO) en la forma original de la arquitectura de la aplicación basada en microservicios.
2- Si el valor de SGM está entre 0.61 y 1, según los valores de los umbrales definidos en la literatura para esta métrica, mientras más cerca de 1 se encuentre el valor de la métrica, más granular es el microservicio; esto quiere decir que es demasiado pequeño en cuanto a cantidad de operaciones (ver Tabla 6). Por lo tanto, se procede a modificar la arquitectura del microservicio buscando acoplarlo (teniendo en cuenta no afectar la cohesión del microservicio resultante) con algún otro microservicio de la aplicación y entonces determinar las métricas con la arquitectura modificada.
3- Si el valor de SGM está entre 0 y 0.2, según los valores de los umbrales definidos en la literatura para esta métrica; mientras más cerca de 0 se encuentre el valor de la métrica, menos granular es el microservicio; esto quiere decir que es demasiado grande en cuanto a cantidad de operaciones (ver Tabla 6). Por lo tanto, se procede a modificar la arquitectura del microservicio buscando dividir el microservicio de la aplicación y entonces determinar las métricas con la arquitectura modificada.
4- Si los valores de las métricas están dentro de los umbrales definidos en la Tabla 6 se aplicará arbitrariamente uno de los dos escenarios (2 o 3) anteriores.
5- Se calcula el promedio de los valores de SGM, LCOM y NOO para la aplicación completa, teniendo en cuenta el número de microservicios que compone la aplicación en ambas

arquitecturas (original y modificada). Es decir, se calcula la media de los valores obtenidos para cada atributo de calidad en cada microservicio que compone la aplicación como se muestra en las ecuaciones (6),(7) y (8). Los valores obtenidos se utilizaron para realizar una comparación entre ellos y así determinar el comportamiento de los atributos de calidad cohesión y complejidad en el balance de atributos de calidad.

$$ALCOM = \frac{\sum_{i=1}^{n} LCOM}{NS} \quad (6)$$

Donde:
NS: cantidad de microservicios
ALCOM: Promedio de falta de cohesión

$$ASGM = \frac{\sum_{i=1}^{n} SGM}{NS} \quad (7)$$

Donde:
NS: cantidad de microservicios
ASGM: Promedio de granularidad en servicios

$$ANOO = \frac{\sum_{i=1}^{n} NOO}{NS} \quad (8)$$

Donde:
NS: cantidad de microservicios
ANOO: Promedio de número de operaciones

El proceso de ejecución de pruebas se representa mediante un diagrama de flujo que se muestra en la Figura 5.

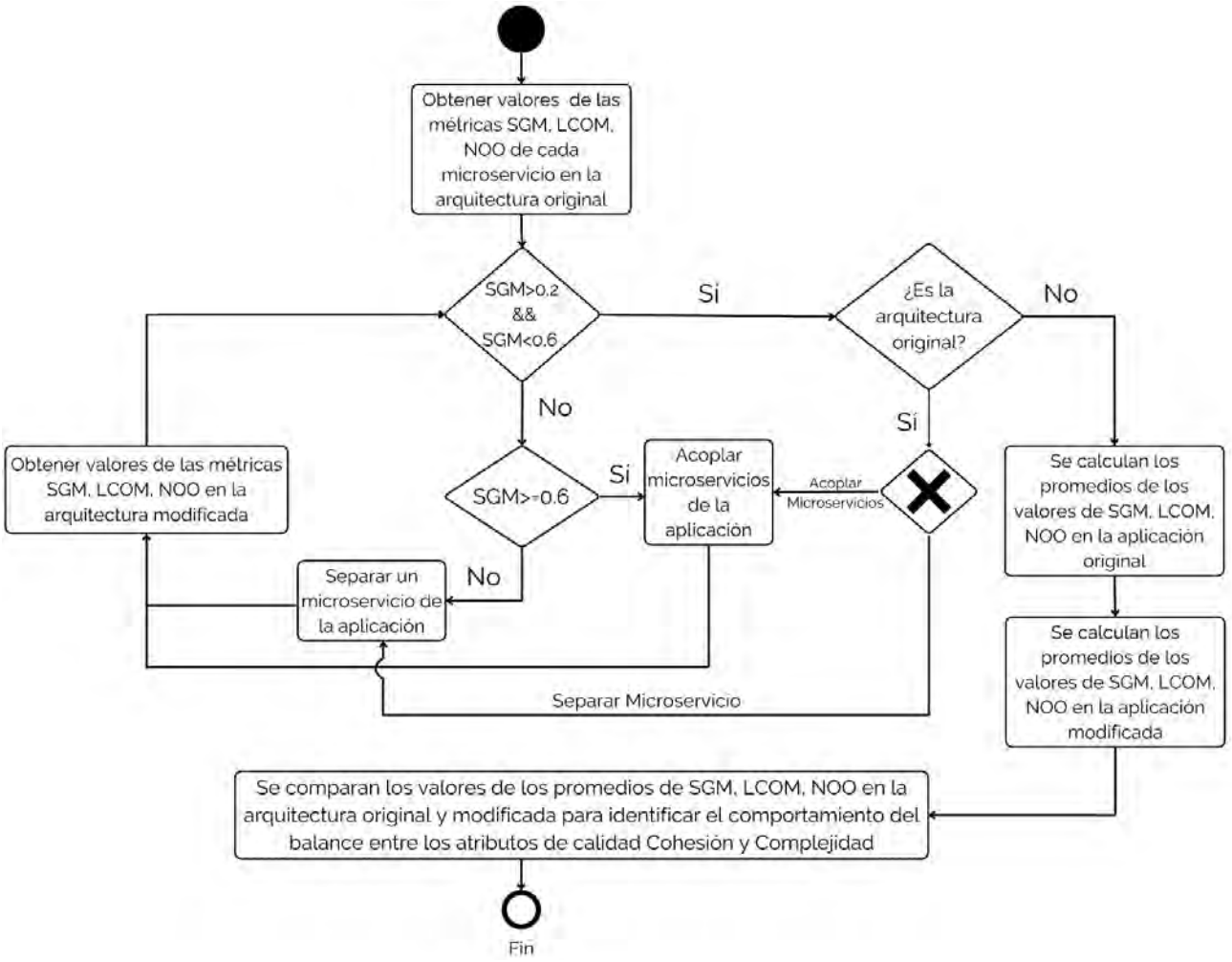

Figura 5. Diagrama de flujo para la ejecución de las pruebas

Las pruebas se ejecutaron en aplicaciones basadas en microservicios Train Ticket, E-Commerce y Transfer-Money. Estas aplicaciones se encuentran descritas en la Tabla 7. Para cada aplicación se calcularon y obtuvieron dos veces los valores de las métricas. Una vez en la arquitectura original, y otra en la arquitectura modificada.

Las pruebas del modelo de calidad de la Figura 4 se basaron en el diseño de pruebas que se describe en la Tabla 8.

Tabla 8. Diseño de pruebas

<table>
<tr><th colspan="2">Diseño de Pruebas</th></tr>
<tr><td colspan="2">Objetivo: Obtener valores de las métricas en la aplicación basada en microservicios en su arquitectura original.</td></tr>
<tr><td colspan="2">Características para probar:
Los valores esperados deben estar dentro del intervalo definido para la SGM entre 0 y 1, LCOM entre 0 y 1.
Los valores obtenidos deben estar relacionados con el código evaluado.</td></tr>
<tr><td>Casos de prueba</td><td rowspan="7">Para cada uno de los casos de prueba anteriores se calcularon y obtuvieron los valores de las métricas de la Tabla 6 para hacer así, el balance de atributos de calidad (cohesión y complejidad) relacionados con la granularidad de microservicios.</td></tr>
<tr><td>CP1- Train Ticket, arquitectura original</td></tr>
<tr><td>CP2- Train Ticket, arquitectura modificada</td></tr>
<tr><td>CP3- E-Commerce, arquitectura original</td></tr>
<tr><td>CP4- E-Commerce, arquitectura modificada</td></tr>
<tr><td>CP5- Transfer-Money, arquitectura original</td></tr>
<tr><td>CP6-Transfer-Money, arquitectura modificada</td></tr>
</table>

5.1.8. Bitácora de pruebas

En la Tabla 9 se muestra la plantilla donde se recogerán los datos de los valores obtenidos al aplicar las pruebas en los microservicios.

Tabla 9. Plantilla para la bitácora de pruebas

Bitácora de Pruebas			
Autor	Nombre	Fecha	xx/xx/20xx
Repositorio	Nombre del Repositorio		
Microservicio	Nombre del Microservicio		
Operación	Nombre de la operación en caso que aplique		
Tipo de operación	Tipo de operación en caso que aplique (GET, POST, PUT, DELETE)		
Objetivo	Objetivo de la prueba		
Nombre de la métrica	Entrada	Salida	
Métrica 1	Entradas para la obtención del valor de la métrica	Resultado obtenido al aplicar la fórmula de la métrica	
Observaciones	Observaciones sobre la obtención de los valores de las métricas		

5.1.9. Obtención de valores de las métricas y atributos de calidad

El proceso de extracción de valores de métricas se lleva a cabo de manera semiautomática. El proceso cuenta de dos etapas.

- En la **primera etapa** se obtienen las métricas (*IPR, FP, OPR, CP OT, O, TP, TO, NUP*) en un archivo de Excel. El archivo de Excel lo exporta una herramienta desarrollada para el cálculo de las métricas en la presente investigación. La aplicación está basada en el análisis del código de microservicios implementados en Java. La misma utiliza las reglas de la librería ANTLR de Java.
- En la **segunda etapa** los datos exportados por la herramienta se organizan en documentos Excel que calculan, de manera automática, las métricas SGM, LCOM y NOO.

Para un mejor manejo de los valores asignados a los atributos de calidad, estos fueron normalizados para que todos los datos obtenidos estén en el rango de 0 a 1. Esto facilita el análisis sobre el comportamiento de los atributos de calidad para el balance.

Para el caso del atributo de calidad Granularidad los valores coinciden con el de la métrica obtenida *SGM* y sus valores de aceptación[2] van de 0.2 a 0.6. Los umbrales obtenidos[3] van de 0 a 1.

$$Granularidad = SGM \tag{9}$$

Valores de aceptación para atributo de calidad Granularidad: 0.2 - 0.6.

En el caso del atributo de calidad Cohesión se le restó a 1 el valor obtenido de la métrica *LCOM* como se muestra en la ecuación (10). Esta transformación se debe realizar para que los valores de las tres mediciones (LCOM, SGM y NOO) tengan la misma dirección. Los valores de aceptación de la métrica *LCOM* son de 0 a 0.8, es decir, que a partir de 0.8 se considera un microservicio con muy baja cohesión. Este razonamiento conduce a que, para el caso del atributo de calidad cohesión, será un microservicio muy poco cohesivo aquel que tenga valores de cohesión entre 0 y 0.19, siendo el valor ideal 1. Un ejemplo de este caso puede ser un microservicio que cuente con una sola operación.

$$Cohesión = 1 - LCOM \tag{10}$$

Valores de aceptación para atributo de calidad Granularidad: 0.21 – 1.

En el caso del atributo de calidad complejidad se le asigna el valor de 1 para el caso de aquellos microservicios que tengan 10 o más operaciones atendiendo a que los valores de aceptación considerados en la presente investigación son de 1 a 9, teniendo en cuenta que valores cercanos o iguales a 1 significa que estamos en presencia de un microservicio complejo. Para asegurar que los

[2] Valores de granularidad más cercanos a 0 significa que el servicio es menos granular o más grande en cuanto a cantidad de operaciones se refiere y más cercanos a 1significa que el servicio es más granular o más pequeño en cuanto a cantidad de operaciones.

[3] Hubo microservicios que los valores obtenidos fueron mayor que 1, para ajustar se dejaron en 1 significando que es un microservicio lo más granular posible (pequeño) atendiendo a la cantidad de operaciones.

valores del atributo de calidad complejidad estén entre 0 y 1 se divide entre 10 el valor de la métrica *NOO*.

$$Complejidad = \begin{cases} NOO/10, & NOO \leq 10 \\ 1, & NOO > 10 \end{cases} \quad donde\ NOO \in \mathbb{N} \tag{11}$$

Valores de aceptación para atributo de calidad Granularidad: 0.1 - 0.9.

En la Tabla 9 se describe el resumen de los valores de aceptación para los atributos de calidad granularidad, cohesión y complejidad.

Tabla 10. Umbrales/Valores de aceptación para atributos de calidad

Atributo de calidad	Umbrales / Valores de aceptación		
Granularidad	**Poco granular**	**Valores aceptados**	**Muy granular**
	0-0.2	0.21 - 0.6	0.61-1
Cohesión	**Poco Cohesivo**	**Valores aceptados**	
	0-0.2	0.21 - 1	
Complejidad	**Valores aceptados**	**Muy complejo**	
	0 - 0.9	0.9-1	

5.2. Casos de prueba

En esta sección se lleva a cabo el procedimiento detallado en la Figura 5. El mismo define los pasos a seguir para la ejecución de las pruebas en los casos de estudio Train Ticket, Transfer-Money y E-Commerce presentados en la Tabla 7.

5.2.1. Caso de estudio Train Ticket

La aplicación Train Ticket [53] es un sistema de reserva de boletos de tren basado en una arquitectura de microservicios que contiene 39 microservicios.

En este caso de estudio se analizaron 10 microservicios pertenecientes a la aplicación Train Ticket. En la Tabla 11, se muestran los valores de DGS y FGS para una operación (*GetAllContacts*) del microservicio *BasicInfo*. Este mismo procedimiento se repite para cada una de las operaciones de cada uno de los microservicios con el objetivo de calcular la métrica SGM. El mismo formato se utiliza

para las métricas LCOM y NOO, en estos casos el procedimiento es para cada microservicio en lugar de para cada operación.

Tabla 11. Bitácora de Prueba Train Ticket. Microservicio Basic Info, operación GetAllContacts

Bitácora de Pruebas			
Autor	Ariel Martínez Roque	Fecha	19/09/2024
Repositorio	Train Ticket		
Microservicio	Basic Info		
Operación	getAllContacts		
Tipo de operación	GET		
Objetivo	Obtener por cada una de las operaciones de un microservicio el valor de las métricas *DGS* (Granularidad de datos de un servicio) y *FGS* (Granularidad funcional de un servicio). Con los valores obtenidos de estas métricas se determina el valor de *SGM* (Métrica de granularidad en servicios).		
Número de métrica	Entrada	Salida	
Métrica 1	*IPR* (Cantidad de parámetros de entrada en una operación), (Total de parámetros de entrada en un microservicio), *OPR* (Cantidad de parámetros de salida en una operación), *CP* (Total de parámetros de salida en un microservicio).	$DGS = \frac{IPR}{\sum_{i=1}^{n} FP} + \frac{OPR}{\sum_{i=1}^{n} CP}$ $DGS = \frac{1}{35} + \frac{1}{20} = 0.08$	
Métrica 2	Valor de *OT* (Peso para una operación específica), *O* (Suma de todos los pesos de las operaciones en un microservicio).	$FGS = \frac{OT}{O}$ $FGS = \frac{1}{50} = 0.02$	
Observaciones	La prueba resultó con valores esperados.		

5.2.1.1. Caso de prueba 1 arquitectura original Train Ticket

En la Figura 3 se muestra la arquitectura original de la aplicación Train Ticket. De color naranja se muestra el microservicio que se dividió atendiendo a los valores de las métricas obtenidas.

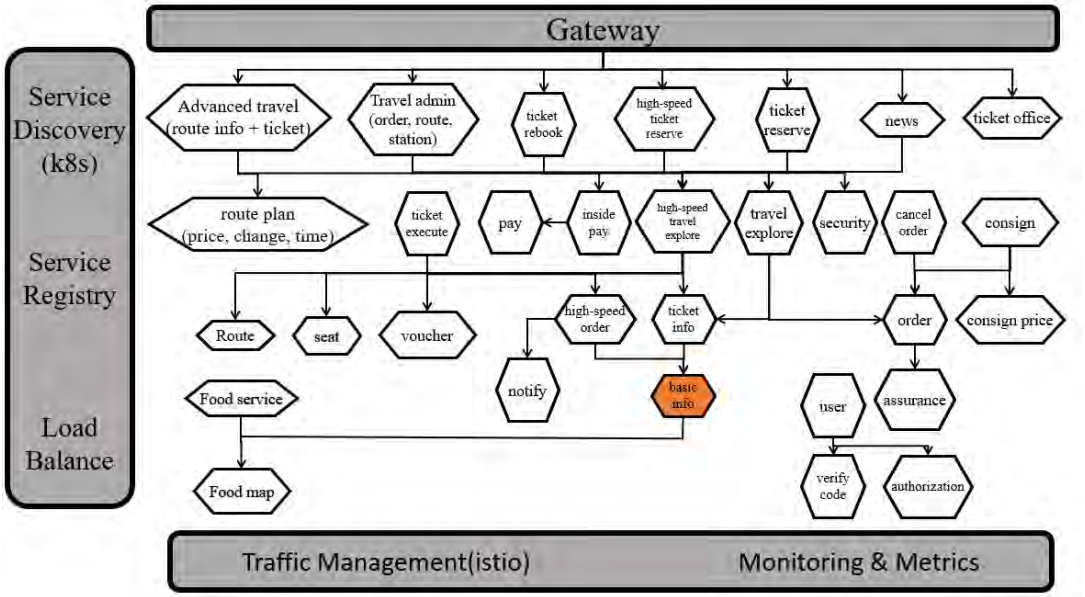

Figura 6. Arquitectura original Train Ticket.
Adaptación de: https://github.com/jo102tz/train-ticket/blob/master/image/2.png

En la Tabla 12 donde se muestran los valores de las métricas para un microservicio de la aplicación Train Ticket, este mismo proceso se realiza para los 9 microservicios restantes de los 10 seleccionados.

Tabla 12. Caso de Prueba 1. Train Ticket, arquitectura original

Caso de Prueba 1. Train Ticket, arquitectura original										
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO/NOO**	**NUP**	**LCOM**
AdminBasicInfo ServiceImpl	getAllContacts	1	1	0.08	0.02	0.1	8	4	4	0.5
	deleteContact	2	1	0.11	0.04					
	modifyContact	2	1	0.11	0.06					
	addContact	2	1	0.11	0.08					
	getAllStations	1	1	0.08	0.02					
	addStation	2	1	0.11	0.08					
	deleteStation	2	1	0.11	0.04					
	modifyStation	2	1	0.11	0.06					
	getAllTrains	1	1	0.08	0.02					
	addTrain	2	1	0.11	0.08					
	deleteTrain	2	1	0.11	0.04					
	modifyTrain	2	1	0.11	0.06					
	getAllConfigs	1	1	0.08	0.02					
	addConfig	2	1	0.11	0.08					
	deleteConfig	2	1	0.11	0.04					
	modifyConfig	2	1	0.11	0.06					
	getAllPrices	1	1	0.08	0.02					
	addPrice	2	1	0.11	0.08					
	deletePrice	2	1	0.11	0.04					
	modifyPrice	2	1	0.11	0.06					

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 13 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad. Estos valores se obtuvieron en 10 de los 39 microservicios que cuenta la aplicación Train Ticket en su arquitectura original.

En color rojo se muestran los valores fuera de los rangos de aceptación o umbrales definidos para cada uno de los atributos de calidad (granularidad: 0-0.2 y de 0.61-1, cohesión: 0-0.2, complejidad 0.9-1).

En color verde se muestran los valores comprendidos dentro de los valores de aceptación o umbrales definidos para cada atributo de calidad (granularidad: 0.2-0.6, cohesión: 0.21-1, complejidad: 0-0.9).

Para más detalles sobre los umbrales o valores de aceptación se aconseja ver la Tabla 6 y la Tabla 10.

Tabla 13. Métricas y atributos de calidad de 10 microservicios de la aplicación Train Ticket [53]

Microservicios	Atributos de Calidad			Métricas		
	Granularidad	Cohesión	Complejidad	SGM	LCOM	NOO
AdminBasicInfo ServiceImpl	0.1	0.2	1	0.1	0.8	20
AdminOrder ServiceImpl	0.5	0.4	0.8	0.5	0.6	8
AdminRoute ServiceImpl	0.7	0.7	0.3	0.7	0.3	3
AdminTravel ServiceImpl	0.5	0.4	0.4	0.5	0.6	4
AdminUser ServiceImpl	0.5	0.5	0.4	0.5	0.5	4
Assurance ServiceImpl	0.3	0.2	0.8	0.3	0.8	8
UserDetails ServiceImpl	1	1	0.1	1	0	1
Token ServiceImpl	1	1	0.1	1	0	1
User ServiceImpl	0.4	0.2	0.5	0.4	0.8	5
Basic ServiceImpl	0.3	0.2	0.6	0.3	0.8	6

En la Figura 7 se muestra la relación existente entre los atributos de calidad Granularidad, Cohesión y Complejidad de 10 microservicios analizados del repositorio Train Ticket [53].

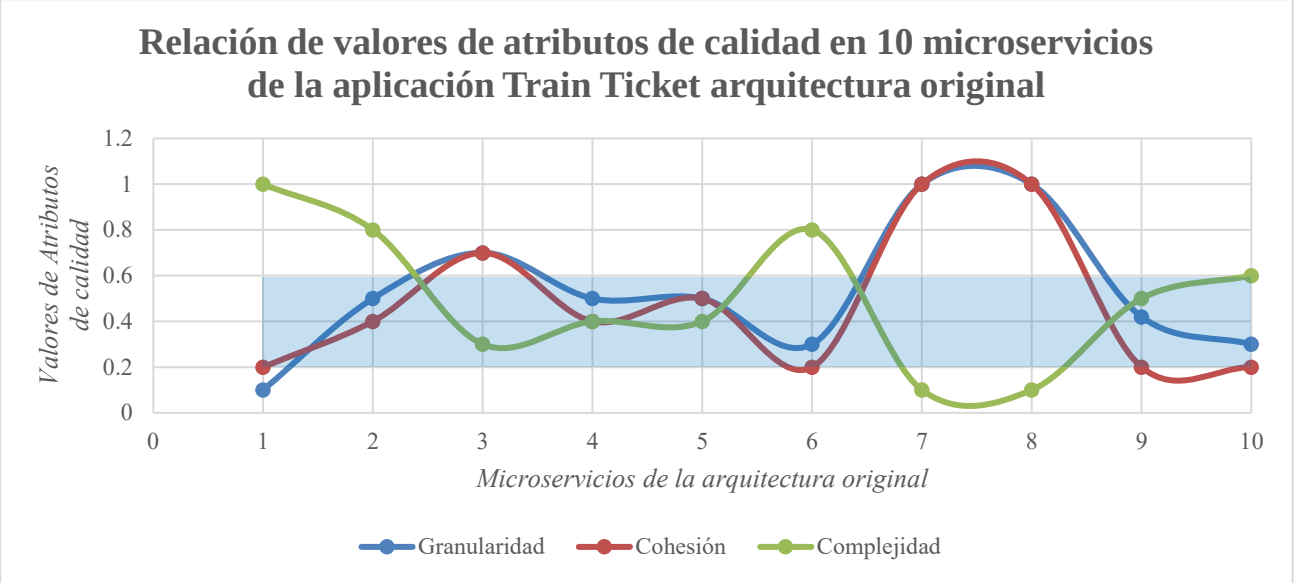

Figura 7. Relación entre Granularidad, Cohesión y Complejidad. Arquitectura original, TrainTicket[53].

En la Figura 7 se muestran, en un gráfico de líneas, los valores de los atributos de calidad determinados para cada uno de los 10 microservicios analizados. La franja de color azul delimita los valores de aceptación o umbrales del atributo de calidad granularidad (0.2-0.6).

En la Figura 7 se puede identificar que en los microservicios 1, 3, 7 y 8; los valores del atributo de calidad granularidad están fuera del umbral definido (0.2-0.6). A continuación, se muestran los microservicios con la cantidad de operaciones en la Tabla 14.

Tabla 14. Microservicios con valores de atributo de calidad granularidad fuera de los umbrales definidos. Train Ticket, arquitectura original

No. en la gráfica de la Figura 7	Nombre del microservicio	Cantidad de Operaciones
1	AdminBasicInfoServiceImpl	20
3	AdminRouteServiceImpl	3
7	UserDetailsServiceImpl	1
8	TokenServiceImpl	1

Analizando los valores de la Tabla 14 se puede concluir que el microservicio *AdminBasicInfoServiceImpl* es muy poco granular, es decir, demasiado grande en cuanto a cantidad de operaciones se refiere. Teniendo en cuenta que los umbrales definidos el microservicio *AdminBasicInfoServiceImpl* es un candidato para quitarle operaciones y generar nuevos

microservicios. En la Tabla 14 se muestra el microservicio *AdminBasicInfoServiceImpl* con todas sus operaciones en su versión original.

Tabla 15. Microservicio AdminBasicInfoServiceImpl con sus operaciones en la arquitectura original Train Ticket

Microservicio	Operaciones
AdminBasicInfoServiceImpl	getAllContacts
	deleteContact
	modifyContact
	addContact
	getAllStations
	addStation
	deleteStation
	modifyStation
	getAllTrains
	addTrain
	deleteTrain
	modifyTrain
	getAllConfigs
	addConfig
	deleteConfig
	modifyConfig
	getAllPrices
	addPrice
	deletePrice
	modifyPrice

5.2.1.2. Caso de prueba 2 arquitectura modificada Train Ticket.

El microservicio *AdminBasicInfoServiceImpl* se dividió en 5 microservicios ***Contacts, Stations, Trains, Configs y Prices***. Los microservicios resultantes de la división propuesta en la nueva arquitectura y sus operaciones se muestran a continuación en la Tabla 16.

.

Tabla 16. Microservicios con sus operaciones resultantes de la división propuesta para el microservicio AdminBasicInfoImpl. Arquitectura modificada Train Ticket.

Microservicio	Operaciones
AdminBasicInfoServiceConfigsImpl	getAllContacts deleteContact modifyContact addContact
AdminBasicInfoServiceContactsImpl	getAllStations addStation deleteStation modifyStation
AdminBasicInfoServicePricesImpl	getAllTrains addTrain deleteTrain modifyTrain
AdminBasicInfoServiceStationsImpl	getAllConfigs addConfig deleteConfig modifyConfig
AdminBasicInfoServiceTrainsImpl	getAllPrices addPrice deletePrice modifyPrice

En la Figura 8 se muestra como quedaría la arquitectura modificada, en verde se muestran los nuevos microservicios generados a partir de la división, en este caso del microservicio *BasicInfo.*

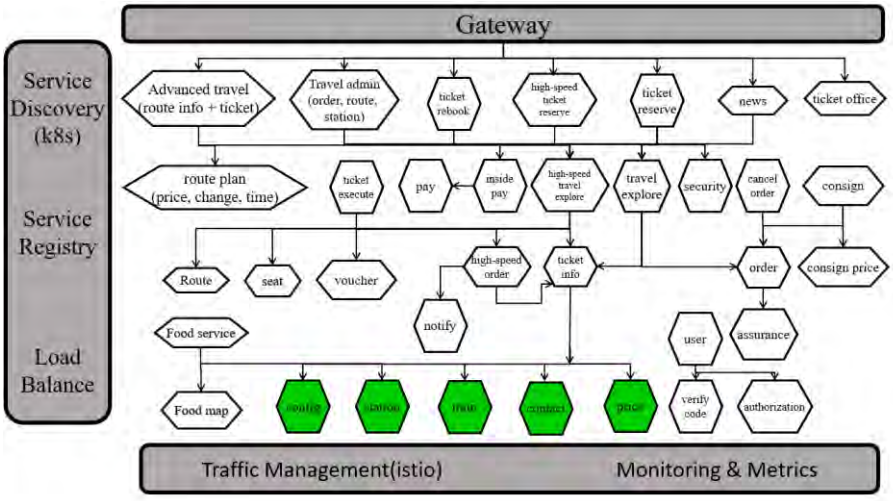

Figura 8. Arquitectura Modificada. Train Ticket

Una vez dividido el microservicio *AdminBasicInfoServiceImpl* se obtienen los valores de las métricas en lo nuevos microservicios generados. Estos valores se muestran en la Tabla 17.

Tabla 17. Caso de prueba 2. Train Ticket, arquitectura modificada

Caso de Prueba 2. Train Ticket, arquitectura modificada										
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO**	**NUP**	**LCOM**
AdminBasicInfo ServiceContacts	getAllContacts	1	1	0.39	0.10	0.52	11	4	4	0.3
	deleteContact	2	1	0.54	0.20					
	modifyContact	2	1	0.54	0.30					
	addContact	2	1	0.54	0.40					
AdminBasicInfo ServiceStations	getAllStations	1	1	0.39	0.10	0.52	11	4	2	0
	addStation	2	1	0.54	0.40					
	deleteStation	2	1	0.54	0.20					
	modifyStation	2	1	0.54	0.30					
AdminBasicInfo ServiceTrains	getAllTrains	1	1	0.39	0.10	0.52	11	4	3	0.1
	addTrain	2	1	0.54	0.40					
	deleteTrain	2	1	0.54	0.20					
	modifyTrain	2	1	0.54	0.30					
AdminBasicInfo ServiceConfigs	getAllConfigs	1	1	0.39	0.10	0.52	11	4	3	0.1
	addConfig	2	1	0.54	0.40					
	deleteConfig	2	1	0.54	0.20					
	modifyConfig	2	1	0.54	0.30					
AdminBasicInfo ServicePrices	getAllPrices	1	1	0.39	0.10	0.52	11	4	2	0
	addPrice	2	1	0.54	0.40					
	deletePrice	2	1	0.54	0.20					
	modifyPrice	2	1	0.54	0.30					

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 18 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad. Estos valores fueron obtenidos en los nuevos microservicios generados a partir de la división del microservicio *AdminBasicInfoServiceImpl*. Los microservicios resultantes de la división forman parte de la arquitectura modificada de la aplicación Train Ticket.

Tabla 18. Valores obtenidos después de dividir el microservicio AdminBasicInfo

Microservicios	Atributos de Calidad			Métricas		
	Granularidad	Cohesión	Complejidad	SGM	LCOM	NOO
AdminBasicInfoService ConfigsImpl	0.5	0.9	0.4	0.5	0.1	4
AdminBasicInfoService ContactsImpl	0.5	0.7	0.4	0.5	0.3	4
AdminBasicInfoService PricesImpl	0.5	1	0.4	0.5	0	4
AdminBasicInfoService StationsImpl	0.5	1	0.4	0.5	0	4
AdminBasicInfoService TrainssImpl	0.5	0.9	0.4	0.5	0.1	4

Luego de la separar las operaciones del microservicio en otros nuevos todos los valores de los atributos de calidad están dentro de los umbrales definidos. En la Figura 9 se muestra un gráfico de líneas de la arquitectura modificada.

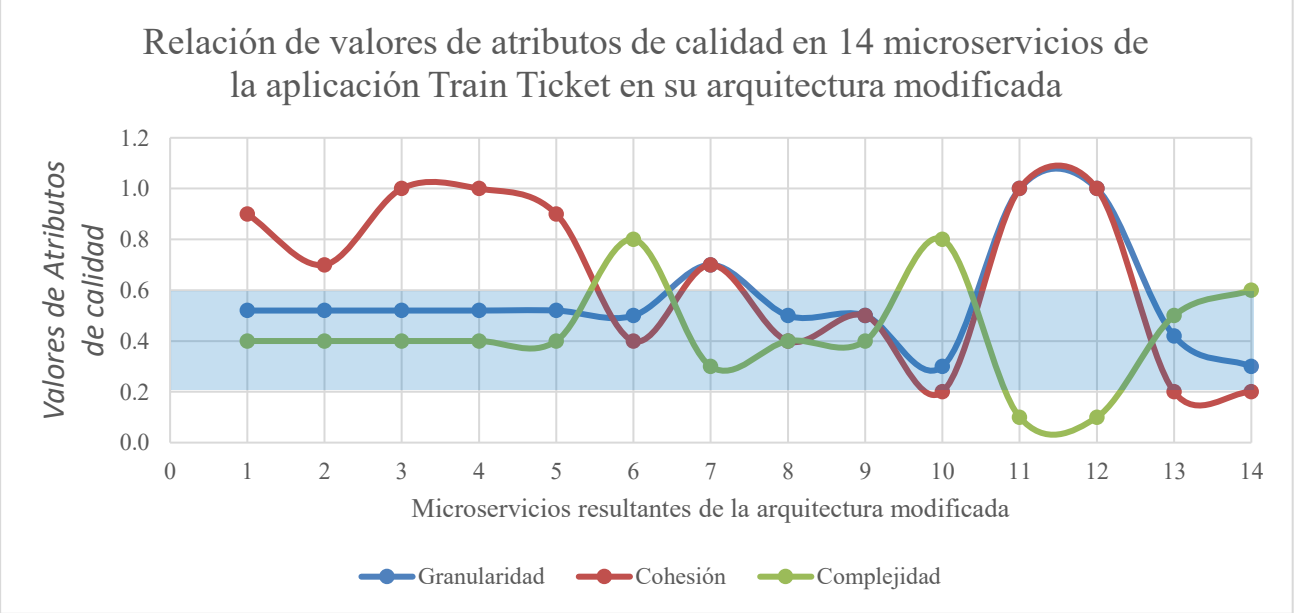

Figura 9. Relación de valores de granularidad, cohesión y complejidad en 10 microservicios de la aplicación Train Ticket en su arquitectura modificada

Se puede observar que los valores de complejidad y granularidad están entre los valores de aceptación (0-0.9) y (0.2-0.6) respectivamente.

Para obtener valores de la aplicación completa se procede a determinar el promedio de los valores de las métricas obtenidas por cada uno de los microservicios que la componen. Los valores serán calculados siguiendo las ecuaciones (6), (7) y (8).

En la Tabla 18 se muestran los valores promedio (ASGM, ALCOM y ANOO) en cada uno de los escenarios (Arquitectura Original y Arquitectura Modificada).

Tabla 19. Promedio de los valores de los atributos de calidad en la arquitectura original y la modificada, Train Ticket

Arquitectura	Granularidad	Cohesión	Complejidad	Cambios
Arquitectura Original	0.53	0.48	0.50	-
Arquitectura Modificada	0.56	0.65	0.43	División del Microservicio Admin Basic Info en 5 Microservicios (Configs, Contacts, Prices, Station, Trains)

En la Figura 10 se muestran los valores a través de una gráfica de barras.

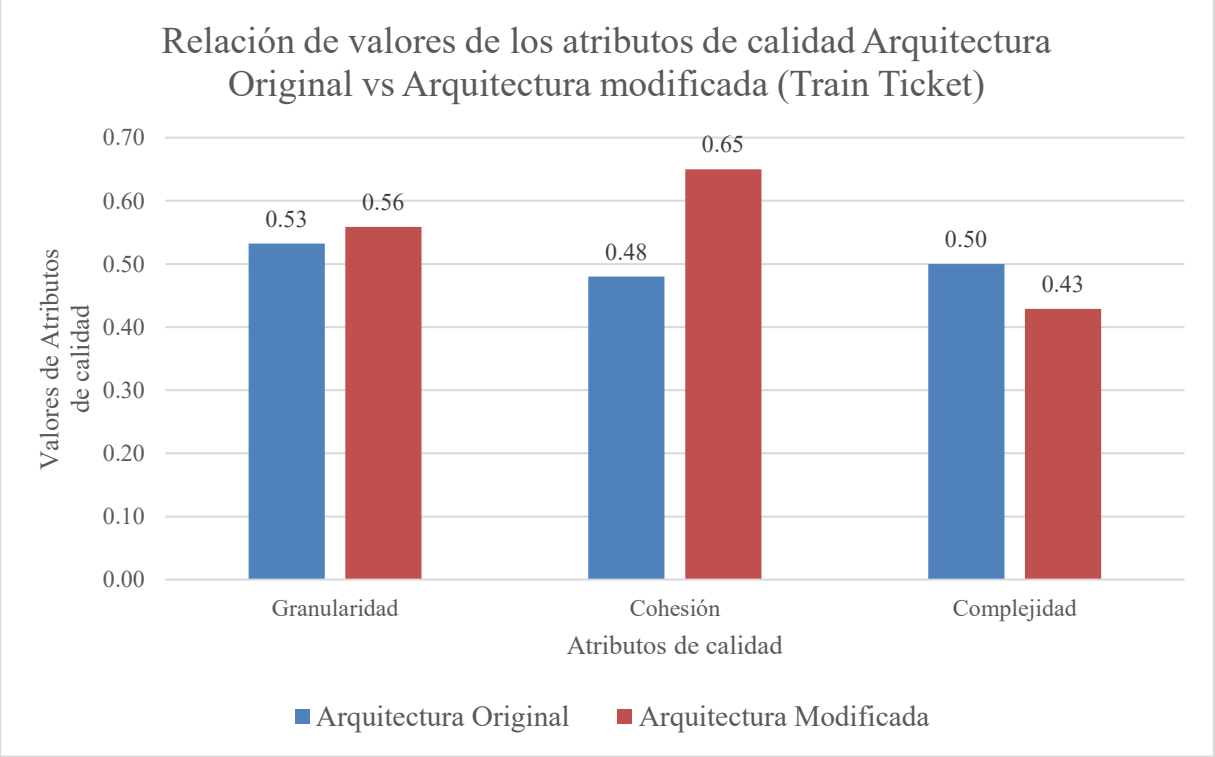

Figura 10. Resumen de valores de atributos de calidad, Train Ticket

5.2.1.3. Análisis de CP1 y CP2

Los casos de prueba "***CP1****- Train Ticket, arquitectura original*" y "***CP2****- Train Ticket, arquitectura modificada"* se ejecutan con datos que arrojan las métricas seleccionadas en el modelo de calidad para el balance de atributos de calidad cohesión y complejidad (ver Figura 4). Estas métricas se obtienen de microservicios que conforman la aplicación Train Ticket.

Para el caso de prueba uno; que consistió en obtener las métricas de, en este caso, de 10 microservicios de los 39 que conforman la aplicación; luego de recopilar los valores de las métricas y de los atributos de calidad se decidió dividir uno de los microservicios *AdminBasicInfoServiceImpl* en 5 microservicios debido a los valores del atributo de calidad granularidad (0.1) en este caso.

Luego de separar las operaciones del microservicio en otros nuevos microservicios se obtienen los valores de las métricas nuevamente. El resumen de los valores obtenidos se muestra en la Figura 10. De este gráfico se puede concluir que, para este caso de estudio, cuando se disminuyó la cantidad de operaciones por microservicio el valor de la granularidad aumentó, la cohesión aumentó y la complejidad disminuyó. En este caso se puede observar además que los atributos de calidad cohesión y complejidad se contraponen.

Se puede concluir que en este caso los atributos granularidad y cohesión tienen un comportamiento directamente proporcional mientras que el atributo de calidad complejidad tiene una relación inversamente proporcional a los atributos de calidad granularidad y cohesión. Esta relación se hace más notable cuando los valores están fuera de los umbrales de aceptación o umbrales definidos en la Tabla 10.

5.2.2. Caso de estudio E-Commerce

E-Commerce [55] es una implementación de una aplicación basada en Microservicios REST en un E-Commerce con Spring boot, Cloud y múltiples módulos. Está conformada por 6 microservicios.

En esta sección se analizaron los 6 microservicios que componen la aplicación E-Commerce. En la Tabla 20, se muestran los valores de SGM, LCOM y NOO para el microservicio *Product.* Este mismo procedimiento se repite para cada uno de los microservicios de la aplicación E-Commerce.

Tabla 20. Bitácora de pruebas. Microservicio Product de la aplicación E-Commerce

Bitácora de pruebas			
Autor	Ariel Martínez Roque	Fecha	9/10/2024
Repositorio	E-Commerce		
Microservicio	Product		
Objetivo	Obtener el valor de las métricas *SGM* (Métrica de granularidad de servicio), *LCOM* (Falta de Cohesión) y *NOO* (Número de operaciones).		
Número de métrica	Entrada	Salida	
Métrica 1	Valor de *DGS* (Granularidad de datos de servicio), *FGS* (Granularidad funcional de un servicio)	$SGM = \sum_{1}^{n} DGS * FGS$ $SGM = 0.34$	
Métrica 2	Valor de *TP* (Total de parámetros en un microservicio), *TO* (Total de operaciones en un microservicio), *NUP* (Número único de parámetros en un microservicio)	$LCOM = 1 - \frac{TP}{TO * NUP}$ $LCOM = 1 - \frac{10}{6 * 10} = 0.8$	
Métrica 3	*NOO* (Cantidad de Operaciones)	$NOO = 6$	
Observaciones	La prueba resultó con valores esperados.		

5.2.2.1. Caso de prueba 3 arquitectura original E-Commerce

En la Figura 11 se muestra la arquitectura original de la aplicación E-Commerce .

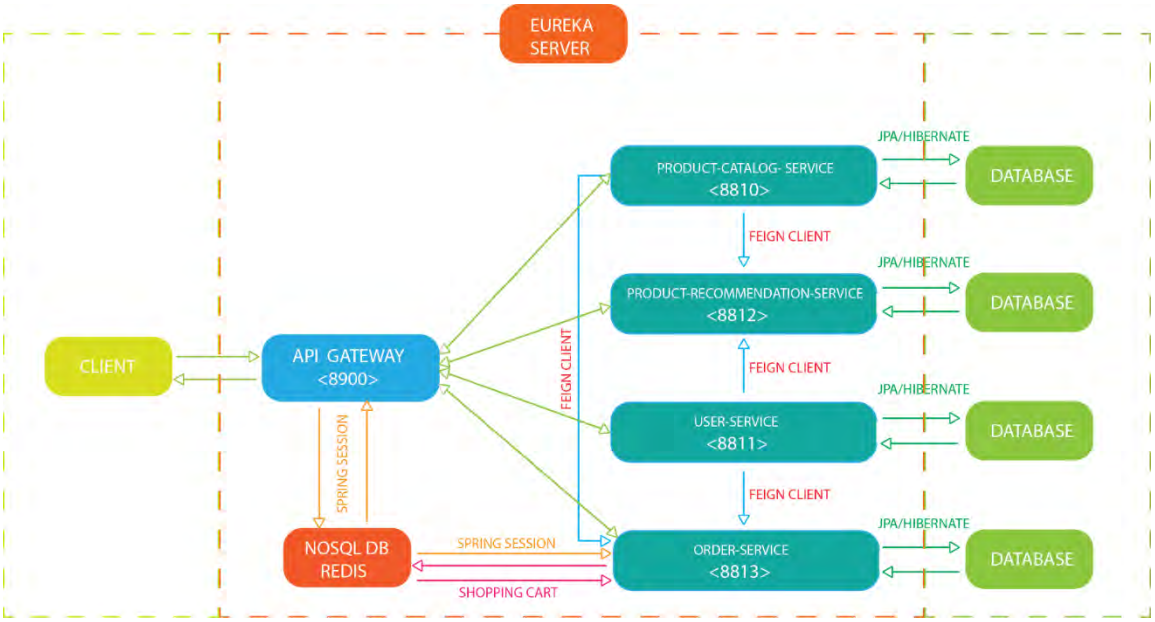

Figura 11. Arquitectura Original, E-Commerce.
Obtenida de: https://user-images.githubusercontent.com/50141193/58799788-845b1c00-8606-11e9-924b-1b4c03a9091c.png

En la Tabla 21 se muestran los valores de las métricas para los microservicios de la aplicación E-Commerce en su arquitectura original.

Tabla 21. Caso de prueba 3. E-Commerce, arquitectura original

Caso de prueba 3. E-Commerce, arquitectura original										
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO**	**NUP**	**LCOM**
CartRedis RepositoryImpl	addItemToCart	2	0	0.29	0.44	0.37	8	4	4	0.5
	getCart	2	1	1.29	0.11					
	deleteItemFromCart	2	0	0.29	0.22					
	deleteCart	1	0	0.14	0.22					
CartServiceImpl	addItemToCart	3	0	0.23	0.29	0.24	17	7	7	0.7
	getCart	1	1	0.33	0.07					
	changeItemQuantity	3	0	0.23	0.21					
	deleteItemFromCart	2	0	0.15	0.14					
	checkIfItemIsExist	2	2	0.65	0.07					
	getAllItemsFromCart	1	1	0.33	0.07					
	deleteCart	1	0	0.08	0.14					
Order ServiceImpl	saveOrder	1	1	2.00	1.00	1	2	1	2	0
Product ServiceImpl	getAllProduct	0	1	0.20	0.10	0.34	10	6	10	0.8
	getAllProduct ByCategory	1	1	0.40	0.10					
	getProductById	1	1	0.40	0.10					
	getAllProductsByName	1	1	0.40	0.10					
	addProduct	1	1	0.40	0.40					
	deleteProduct	1	0	0.20	0.20					
Recommendation ServiceImpl	saveRecommendation	1.0	1.0	0.6	0.5	0.5	7	4	7	0.8
	getAllRecommendation ByProductName	1.0	1.0	0.6	0.1					
	deleteRecommendation	1.0	0.0	0.3	0.3					
	getRecommendationBy Id	1.0	1.0	0.6	0.1					
UserService	getAllUsers	0.0	1.0	0.3	0.1	0.54	7	4	7	0.8
	getUserById	1.0	1.0	0.6	0.1					
	getUserByName	1.0	1.0	0.6	0.1					
	saveUser	1.0	1.0	0.6	0.6					

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 22 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad. Estos valores fueron obtenidos en todos los microservicios que conforman la aplicación E-Commerce en su arquitectura original.

En color rojo se muestran los valores fuera de los valores de aceptación o umbrales definidos para cada uno de los atributos de calidad (granularidad: 0-0.2 y de 0.61-1, cohesión: 0-0.2, complejidad 0.9-1).

En color verde se muestran los valores comprendidos dentro de los valores de aceptación o umbrales definidos para cada atributo de calidad (granularidad: 0.2-0.6, cohesión: 0.21-1, complejidad: 0-0.9).

Para más detalles sobre los umbrales o valores de aceptación se aconseja ver la Tabla 6 y la Tabla 10.

Tabla 22. Métricas y atributos de calidad de los microservicios la aplicación E-Commerce

Microservicios	Atributos de Calidad			Métricas		
	Granularidad	Cohesión	Complejidad	SGM	LCOM	NOO
CartRedisRepositoryImpl	0.4	0.5	0.4	0.4	0.5	4
CartServiceImpl	0.2	0.3	0.7	0.2	0.7	7
OrderServiceImpl	1.0	1	0.1	1.0	0	1
ProductServiceImpl	0.3	0.2	0.6	0.3	0.8	6
RecommendationServiceImpl	0.5	0.2	0.4	0.5	0.8	4
UserService	0.5	0.2	0.4	0.5	0.8	4

A continuación se muestra la relación existente entre los atributos de calidad Granularidad, Cohesión y Complejidad de 6 microservicios analizados del repositorio E-Commerce [55] en la Figura 12.

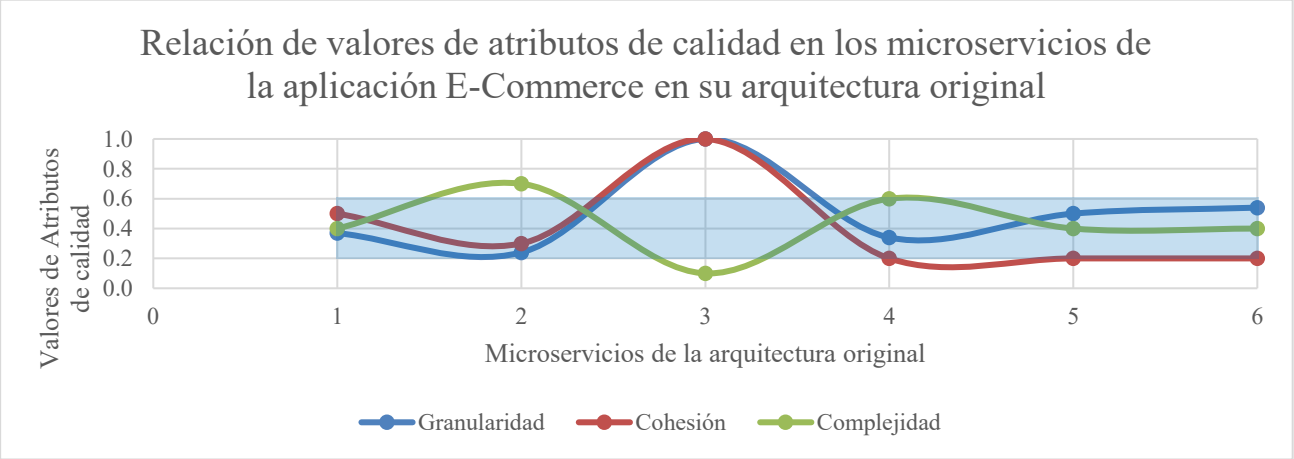

Figura 12. Relación entre Granularidad, Cohesión y Complejidad. Arquitectura original, E-Commerce

En la Figura 12 se muestran los valores de los atributos de calidad determinados para cada uno de los 6 microservicios de la aplicación E-Commerce. La franja de color azul delimita los valores de aceptación o umbrales del atributo de calidad granularidad (0.2-0.6).

En la Figura 12 se puede identificar que en el microservicio 3 el valor del atributo de calidad granularidad está fuera del umbral definido (0.2-0.6). En este caso el valor del atributo de calidad granularidad para el microservicio *Order* es igual a 1. Este valor significa que el microservicio es demasiado granular. En términos de número de operaciones quiere decir que tiene muy pocas, para este caso específico la cantidad de operaciones que tiene el microservicio *Order* (microservicio 3 en la Figura 12). En la Tabla 23 se muestran los microservicios con la cantidad de operaciones.

Tabla 23. Microservicios con valores de atributo de calidad granularidad fuera de los umbrales definidos. E-Commerce, arquitectura original

No. en la gráfica de la Figura 12	Nombre del microservicio	Cantidad de Operaciones
3	OrderServiceImpl	1

Se puede concluir que el microservicio *OrderService* es muy granular ($Granularidad = 1$). Esto quiere decir que es un microservicio demasiado pequeño en cuanto a número de operaciones ($NOO = 1$) teniendo en cuenta los umbrales definidos para granularidad (0.2-0.6). Por lo tanto, es un microservicio candidato para acoplar.

Otros candidatos para acoplar con el microservicio *OrderService* teniendo en cuenta el valor de granularidad es el microservicio *UserService* (0.5) que está dentro de los umbrales definidos (0.2-0.6) y aún tiene margen para poder cambiar dicho valor y aún permanecer dentro de los valores de aceptación. En el mismo caso se encuentra el microservicio *RecommendationServiceImpl*.

De los otros microservicios hay tres que son poco cohesivos, de estos hay uno que posee parámetros similares al microservicio *OrderService*, en este caso es el microservicio *UserService*.

Atendiendo a los valores obtenidos en el análisis de los microservicios y mostrados en la Tabla 22 y la lógica de negocio de la aplicación, se procede a acoplar los microservicios *OrderService* y *UserService*.

En la Tabla 24, se muestran los microservicios que se acoplan y sus correspondientes operaciones.

Tabla 24. Microservicios para acoplar con sus operaciones en arquitectura original E-Commerce

Microservicio	Operaciones
OrderServiceImpl	saveOrder
UserServiceImpl	getAllUsers
	getUserById
	getUserByName
	saveUser

5.2.1.1. Caso de prueba 4 arquitectura modificada E-Commerce

Luego de acoplar los microservicios *OrderService* y *UserService* se obtiene el microservicio *User-Order-Service* mostrado a continuación con sus operaciones en la Tabla 25.

Tabla 25. Nuevo microservicio resultante con sus operaciones en la arquitectura modificada E-Commerce

Microservicio	Operaciones
User-Order-Service	saveOrder
	getAllUsers
	getUserById
	getUserByName
	saveUser

En la Figura 13 se muestra como quedaría la arquitectura modificada con la unión de los microservicios *User* y *Order.* De color blanco se muestra el nuevo microservicio generado.

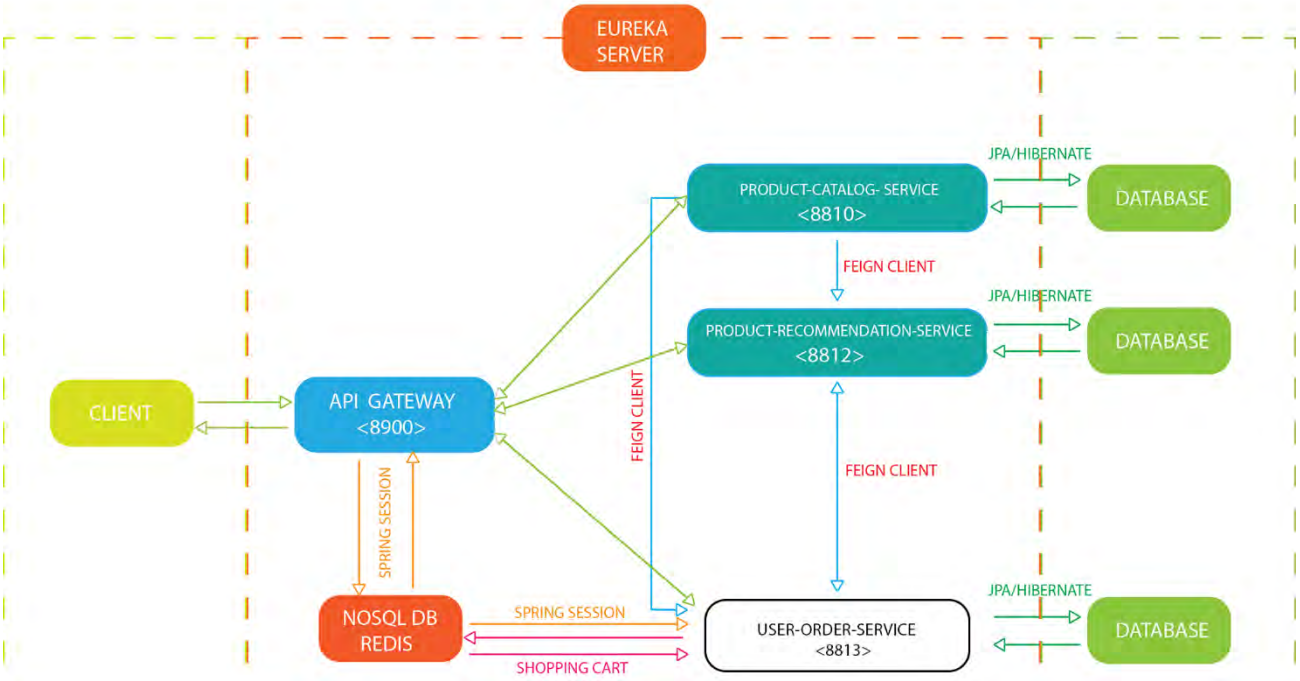

Figura 13. Arquitectura Modificada E-commerce.

En la Tabla 26 se muestran los datos de métricas obtenidos de los microservicios de la aplicación E-Commerce luego de modificar su arquitectura. Es válido recordar que la modificación consistió en acoplar los microservicios *Order* y *User*.

Tabla 26. Obtención de valores de las métricas del caso de estudio E-Commerce. Arquitectura modificada

Caso de prueba 4. E-Commerce, arquitectura modificada										
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO**	**NUP**	**LCOM**
CartRedisRepositoryImpl	addItemToCart	2	0	0.29	0.44	0.37	8	4	4	0.5
	getCart	2	1	1.29	0.11					
	deleteItemFromCart	2	0	0.29	0.22					
	deleteCart	1	0	0.14	0.22					
CartServiceImpl	addItemToCart	3	0	0.23	0.29	0.24	17	7	7	0.7
	getCart	1	1	0.33	0.07					
	changeItemQuantity	3	0	0.23	0.21					
	deleteItemFromCart	2	0	0.15	0.14					
	checkIfItemIsExist	2	2	0.65	0.07					
	getAllItemsFromCart	1	1	0.33	0.07					
	deleteCart	1	0	0.08	0.14					
ProductServiceImpl	getAllProduct	0	1	0.20	0.10	0.34	10	6	10	0.8
	getAllProductByCategory	1	1	0.40	0.10					
	getProductById	1	1	0.40	0.10					
	getAllProductsByName	1	1	0.40	0.10					
	addProduct	1	1	0.40	0.40					
	deleteProduct	1	0	0.20	0.20					
Recommendation ServiceImpl	saveRecommendation	1.0	1.0	0.6	0.5	0.5	7	4	7	0.8
	getAllRecommendation ByProductName	1.0	1.0	0.6	0.1					
	deleteRecommendation	1.0	0.0	0.3	0.3					
	getRecommendationById	1.0	1.0	0.6	0.1					
User-Order-Service	saveOrder	1.0	1.0	0.45	0.36	0.43	8	5	8	0.8
	getAllUsers	0.0	1.0	0.2	0.09					
	getUserById	1.0	1.0	0.45	0.09					
	getUserByName	1.0	1.0	0.45	0.09					
	saveUser	1.0	1.0	0.45	0.36					

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 27 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad de los microservicios que componen la arquitectura modificada de la aplicación E-Commerce. Estos valores fueron obtenidos de la nueva arquitectura resultante del acoplamiento de los microservicios *OrderService* y *UserService*.

En la Tabla 27, de color rojo se muestran los valores fuera de los umbrales definidos (granularidad: 0-0.2 y de 0.61-1, cohesión: 0.21-1 y complejidad:0-0.9) y en color verde los valores comprendidos en los umbrales definidos (granularidad: 0.21-0.6, cohesión: 0.21-1 y complejidad:0-0.9).

Tabla 27. Valores de las métricas luego de modificar la arquitectura, E-Commerce

Microservicios	Atributos de Calidad			Métricas		
	Granularidad	Cohesión	Complejidad	SGM	LCOM	NOO
CartRedis RepositoryImpl	0.4	0.5	0.4	0.4	0.5	4
Cart ServiceImpl	0.2	0.3	0.7	0.2	0.7	7
Product ServiceImpl	0.3	0.2	0.6	0.3	0.8	6
Recommendation ServiceImpl	0.5	0.2	0.4	0.5	0.8	4
UserOrder Service	0.4	0.2	0.5	0.4	0.8	5

En la Figura 9 se muestra un gráfico de líneas que relaciona los valores de los atributos de calidad granularidad, cohesión y complejidad de los microservicios resultantes en la arquitectura modificada. En el gráfico de la Figura 9 el microservicio Order-User-Service es el número 5. Se puede apreciar que el valor del atributo de calidad granularidad del microservicio generado (5) está dentro de los umbrales definidos (0.2-0.6).

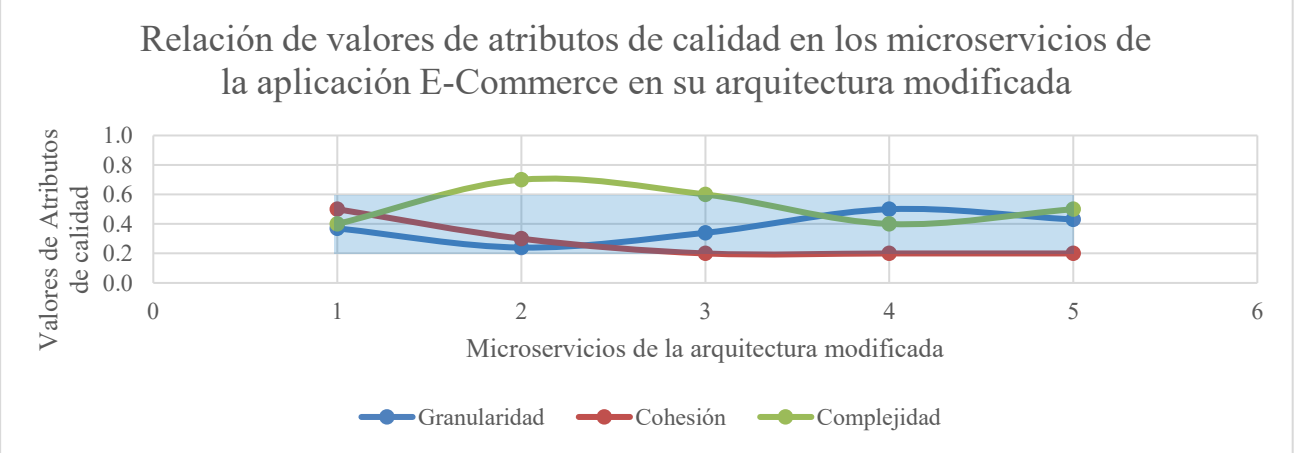

Figura 14. Atributos de calidad luego de la modificación en la arquitectura, E-Commerce

Para obtener valores de los atributos de calidad de la aplicación completa se procede a determinar el promedio de los valores de las métricas obtenidas por cada uno de los microservicios que la componen. Los valores serán calculados siguiendo las ecuaciones (6), (7) y (8).

Para una mejor comprensión del comportamiento de los atributos de calidad ante los cambios en la arquitectura se muestran los valores promedio (ASGM, ALCOM y ANOO) en cada uno de los escenarios (Arquitectura Original y Arquitectura Modificada) en la Tabla 28 a continuación mostrada.

Tabla 28. Promedio de los valores de los atributos de calidad en la arquitectura original y la modificada, E-Commerce

Arquitectura	Granularidad	Cohesión	Complejidad	Cambios
Original	0.50	0.40	0.43	-
Modificada	0.38	0.28	0.52	Acoplamiento de los microservicios User y Order

Seguidamente se representan en una gráfica de barras los valores en la Figura 15.

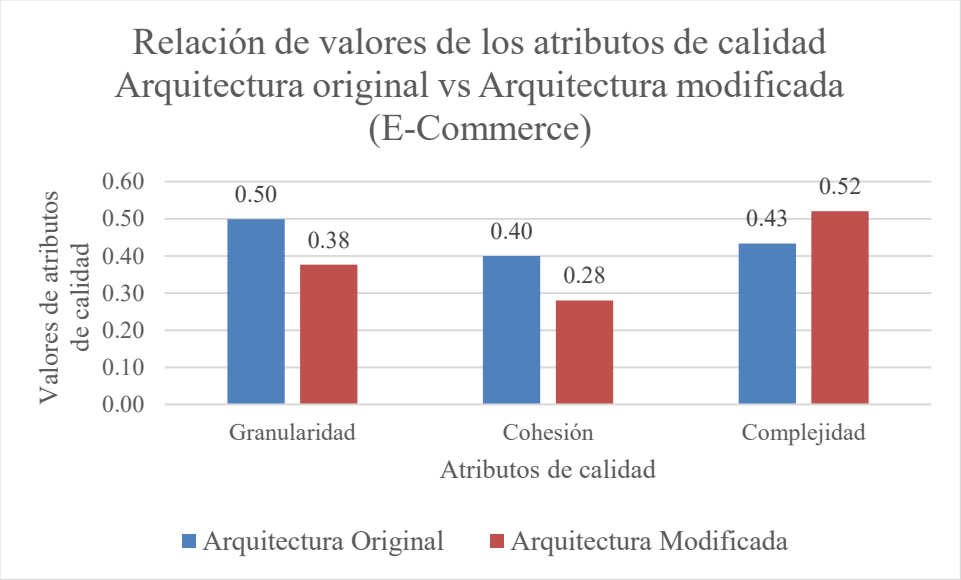

Figura 15. Resumen de valores de los atributos de calidad, E-Commerce

5.2.1.2. Análisis de CP3 y CP4

Las pruebas *"**CP3**- E-Commerce, arquitectura original" y "**CP4**- E-Commerce, arquitectura modificada"* se ejecutan con datos que arrojan las métricas seleccionadas en el modelo de calidad para el balance de atributos de calidad seleccionados en el mismo (ver Figura 4). Estas métricas se obtienen de microservicios que integran la aplicación E-Commerce.

Para el caso de prueba tres, que consistió en obtener las métricas de todos los microservicios (6) que conforman la aplicación, luego de recopilar los valores de las métricas y de los atributos de calidad se

decidió acoplar dos de los microservicios analizados debido a los valores de granularidad. En este caso se identificó el microservicio *Order* con una sola operación.

Luego de acoplar los microservicios (*User* y *Order*) se repite el proceso de obtención de los valores de las métricas. El resumen de los valores obtenidos se muestra en la Figura 15. De este gráfico se puede concluir que, para este caso de estudio (E-Commerce), cuando se aumentó la cantidad de operaciones por microservicio el valor de la granularidad disminuyó, la cohesión disminuyó y la complejidad aumentó. En este caso de estudio se puede observar, al igual que en el anterior, que los atributos de calidad cohesión y complejidad se contraponen.

5.2.2. Caso de Prueba Transfer-Money

Aplicación bancaria cuya arquitectura se basa en microservicios, Event Sourcing y CQRS. Está construida utilizando Spring Cloud, Spring Boot y la plataforma Eventuate. La aplicación se utiliza en laboratorios prácticos que forman parte de una clase de servicios de microservicios impartida por Chris Richardson.

En esta sección se analizaron los 6 microservicios que componen la aplicación Transfer-Money. A continuación, en la Tabla 20, se muestran los valores de SGM, LCOM y NOO para el microservicio *Account.* Este mismo procedimiento se repite para cada uno de los microservicios de la aplicación Transfer-Money.

Tabla 29. Bitácora de pruebas. Microservicio Account de la aplicación Transfer-Money

Bitácora de pruebas			
Autor	Ariel Martínez Roque	Fecha	24/10/2024
Repositorio	Transfer-Money		
Microservicio	Account		
Objetivo	Obtener el valor de las métricas *SGM* (Métrica de granularidad de servicio), *LCOM* (Falta de Cohesión) y *NOO* (Número de operaciones).		
Número de métrica	Entrada	Salida	
Métrica 1	Valor de *DGS* (Granularidad de datos de servicio), *FGS* (Granularidad funcional de un servicio)	$SGM = \sum_{1}^{n} DGS * FGS$ $SGM = 0.18$	
Métrica 2	Valor de *TP* (Total de parámetros en un microservicio), *TO* (Total de operaciones en un microservicio), *NUP* (Número único de parámetros en un microservicio)	$LCOM = 1 - \frac{TP}{TO * NUP}$ $LCOM = 1 - \frac{15}{9 * 7} = 0.76$	
Métrica 3	*NOO* (Cantidad de Operaciones)	$NOO = 9$	
Observaciones	La prueba resultó con valores esperados.		

5.2.2.1. Caso de prueba 5 arquitectura original Transfer-Money

En la Tabla 30 donde se muestran los valores de las métricas para cada uno de los microservicios de la aplicación Transfer-Money en su arquitectura original. Cabe mencionar que, para el caso de esta aplicación, en el repositorio de donde se obtuvo no aparece el diagrama de la arquitectura de la aplicación. En este caso solamente se muestran todos los microservicios que la componen con cada una de sus operaciones.

Tabla 30. Caso de Prueba 5. Transfer-Money, arquitectura original

Caso de Prueba 5 Transfer-Money, arquitectura original										
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO/NOO**	**NUP**	**LCOM**
AccountGroupService	createAccountGroup	1	0	0.10	0.33	0.3	14	6	6	0.6
	addAccount	2	1	0.45	0.33					
	findAccountGroup	1	0	0.10	0.08					
	verifyParentGroupExists	2	3	0.95	0.08					
	noteChildAddedToParent	2	0	0.20	0.08					
	noteParentMissing	2	0	0.20	0.08					
AccountService	openAccount	1	1	0.58	0.06	0.18	15	9	7	0.8
	publishAccount OpenedEvent	1	0	0.08	0.22					
	findAccount	1	1	0.58	0.06					
	noteCustomerValidated	1	0	0.08	0.06					
	noteCustomer ValidationFailed	1	0	0.08	0.06					
	getdebit	2	0	0.15	0.06					
	publishAccount DebitedEvent	2	0	0.15	0.22					
	getcredit	2	0	0.15	0.06					
	publishAccount CreditedEvent	2	0	0.15	0.22					
AccountWith CustomerService	getAccount WithCustomer	1	2	2.00	1.00	1	3	1	3	0.8
CustomerService	createCustomer	1	1	0.75	0.67	0.71	6	3	6	0.7
	findCustomer	1	1	0.75	0.17					
	validateCustomer	2.0	0.0	0.5	0.2					
CustomerViewService	createCustomer	2.0	0.0	0.1	0.4	0.23	24	7	11	0.7
	openAccount	3.0	0.0	0.1	0.1					
	getkeyForBalance	1.0	1.0	0.4	0.1					
	getkeyForChange	2.0	1.0	0.4	0.1					
	getdebitAccount	6.0	0.0	0.3	0.1					
	getcreditAccount	6.0	0.0	0.3	0.1					
	findByCustomerId	1.0	1.0	0.4	0.1					
MoneyTransferService	createMoneyTransfer	1.00	1.00	0.67	0.27	0.38	8	5	6	0.7
	createTransfer MoneySaga	2.00	0.00	0.33	0.27					
	findMoneyTransfer	1.00	1.00	0.67	0.07					
	completeTransfer	1.00	0.00	0.17	0.27					
	cancelTransfer	1.00	0.00	0.17	0.13					

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 31 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad. Estos valores fueron obtenidos en todos los microservicios que conforman la aplicación Transfer-Money en su arquitectura original.

En color rojo se muestran los valores fuera de los valores de aceptación o umbrales definidos para cada uno de los atributos de calidad (granularidad: 0-0.2 y de 0.61-1, cohesión: 0-0.2, complejidad 0.9-1).

En color verde se muestran los valores comprendidos dentro de los valores de aceptación o umbrales definidos para cada atributo de calidad (granularidad: 0.2-0.6, cohesión: 0.21-1, complejidad: 0-0.9).

Para más detalles sobre los umbrales o valores de aceptación se aconseja ver la Tabla 6 y la Tabla 10.

Tabla 31. Valores de los atributos de calidad en la arquitectura original Transfer-Money

Microservicio	**Atributos de Calidad**			**Métricas**		
	Granularidad	**Cohesión**	**Complejidad**	**SGM**	**LCOM**	**NOO**
AccountGroup Service	0.30	0.39	0.60	0.61	6.00	0.30
Account Service	0.18	0.24	0.90	0.76	9.00	0.18
AccountWithCustomer Service	1.00	1.00	0.10	0.00	1.00	1.00
Customer Service	0.71	0.34	0.30	0.66	3.00	0.71
CustomerView Service	0.23	0.32	0.70	0.68	7.00	0.23
MoneyTransfer Service	0.38	0.27	0.50	0.73	5.00	0.38

En la Figura 16 se muestra la relación existente entre los atributos de calidad Granularidad, Cohesión y Complejidad de los 6 microservicios analizados del repositorio Transfer-Money [54].

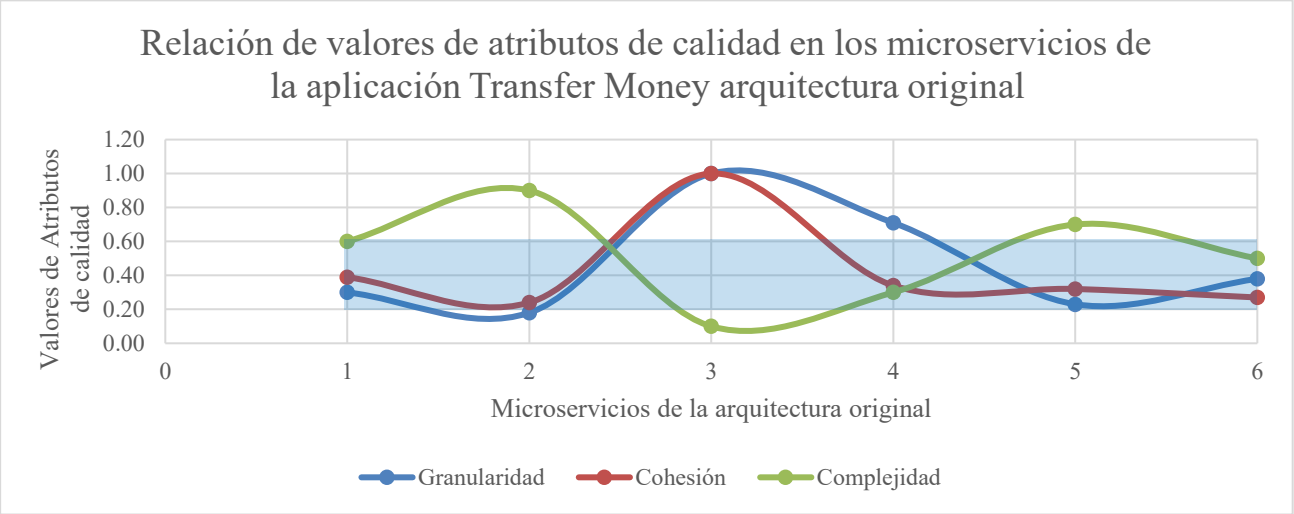

Figura 16. Relación entre los valores de los microservicios en arquitectura original Transfer-Money

En la Figura 16 se muestran los valores de los atributos de calidad determinados para cada uno de los 6 microservicios de la aplicación Transfer-Money. La franja de color azul delimita los valores de aceptación o umbrales del atributo de calidad granularidad (0.2-0.6).

En el gráfico de la Figura 16 se puede observar que los microservicios 3 y 4 tienen valores de granularidad fuera de los valores de aceptación definidos (0.2-0.6). En este caso lo microservicios tienen valores entre 0.6 y 1. Esto quiere decir que son muy pequeños en cuánto a cantidad de operaciones. En este caso para el microservicio 3 (*AccountWithCustomer*) el valor del atributo de calidad granularidad es 1 y para el microservicio 4 (*Customer*) el valor del atributo de calidad granularidad es 0.71. Asimismo, la cantidad de operaciones para los microservicios 3 y 4 es de 1 y 3 respectivamente. en la Tabla 32 se muestran los microservicios con la cantidad de operaciones.

Tabla 32. Microservicios con valores de atributo de calidad granularidad mayor de 0.6. Transfer-Money, arquitectura original

No. en la gráfica de la Figura 12	Nombre del microservicio	Cantidad de Operaciones
3	AccountWithCustomerService	1
4	CustomerService	3

A partir del análisis anterior se decide acoplar los microservicios *AccountWithCustomer* (microservicio 3 en el gráfico de la Figura 16) y *Customer* (microservicio 4 en el gráfico de la Figura 16).

En la Tabla 33, se muestran los microservicios que se acoplan y sus correspondientes operaciones.

Tabla 33. Microservicios para acoplar con sus operaciones en la arquitectura original, Transfer-Money

Microservicio	Operaciones
AccountWithCustomerService	getAccountWithCustomer
CustomerService	createCustomer
	findCustomer
	validateCustomer

5.2.2.2. Caso de prueba 6 arquitectura modificada Transfer-Money

Luego de acoplar los microservicios *AccountWithCustomerService* y *CustomerService* se obtiene el microservicio Customer_AccountWithCustomerService mostrado a continuación con sus operaciones en la Tabla 34.

Tabla 34. Microservicio resultante con sus operaciones en la arquitectura modificada, Train Ticket

Microservicio	Operaciones
Customer_AccountWithCustomerService	getAccountWithCustomer
	createCustomer
	findCustomer
	validateCustomer

En la Tabla 35 se muestran los datos de métricas obtenidos de los microservicios de la aplicación Transfer-Money luego de modificar su arquitectura. Es válido recordar que la modificación consistió en acoplar los microservicios *Customer_AccountWithCustomerService* y *CustomerService*.

Tabla 35. Caso de prueba 6. Transfer-Money, arquitectura modificada

Caso de Prueba 6 Transfer-Money, arquitectura modificada											
Microservicio	**Operaciones**	**IPR**	**OPR**	**DGS**	**FGS**	**SGM**	**TP**	**TO**	**NUP**	**LCOM**	**NOO**
AccountGroupService	createAccountGroup	1	0	0.10	0.33	0.3	14	6	6	0.6	6
	addAccount	2	1	0.45	0.33						
	findAccountGroup	1	0	0.10	0.08						
	verifyParentGroupExists	2	3	0.95	0.08						
	noteChildAddedToParent	2	0	0.20	0.08						
	noteParentMissing	2	0	0.20	0.08						
Account Service	openAccount	1	1	0.58	0.06	0.18	15	9	7	0.8	9
	publishAccountOpened Event	1	0	0.08	0.22						
	findAccount	1	1	0.58	0.06						
	noteCustomerValidated	1	0	0.08	0.06						
	noteCustomerValidation Failed	1	0	0.08	0.06						
	getdebit	2	0	0.15	0.06						
	publishAccountDebited Event	2	0	0.15	0.22						
	getcredit	2	0	0.15	0.06						
	publishAccount CreditedEvent	2	0	0.15	0.22						
Customer_Account With Customer Service	getAccountWith Customer	1	2	0.70	0.14	0.48	9	4	7	0.8	4
	createCustomer	1	1	0.45	0.57						
	findCustomer	1	1	0.45	0.14						
	validateCustomer	2.0	0.0	0.4	0.1						
CustomerView Service	createCustomer	2.0	0.0	0.1	0.4	0.23	24	7	11	0.7	7
	openAccount	3.0	0.0	0.1	0.1						
	getkeyForBalance	1.0	1.0	0.4	0.1						
	getkeyForChange	2.0	1.0	0.4	0.1						
	getdebitAccount	6.0	0.0	0.3	0.1						
	getcreditAccount	6.0	0.0	0.3	0.1						
	findByCustomerId	1.0	1.0	0.4	0.1						
MoneyTransfer Service	createMoneyTransfer	1.00	1.00	0.67	0.27	0.38	8	5	6	0.7	5
	createTransfer MoneySaga	2.00	0.00	0.33	0.27						
	findMoneyTransfer	1.00	1.00	0.67	0.07						
	completeTransfer	1.00	0.00	0.17	0.27						
	cancelTransfer	1.00	0.00	0.17	0.13						

Nota:

IPR: *Parámetros de entrada,* ***OPR:*** *Parámetros de salida,* ***DGS:*** *Granularidad de datos,* ***FGS:*** *Granularidad funcional,* ***SGM:*** *Métrica de granularidad,* ***TP:*** *Total de parámetros,* ***TO/NOO:*** *Total de operaciones/Número de operaciones,* ***NUP:*** *Número único de parámetros,* ***LCOM:*** *Falta de cohesión*

En la Tabla 36 se muestra un resumen de los valores de los atributos de calidad granularidad, cohesión y complejidad de los microservicios que componen la arquitectura modificada de la aplicación

Transfer-Money. Estos valores fueron obtenidos de la nueva arquitectura resultante de acoplar los microservicios *CustomerService* y *CustomerService*.

En la Tabla 36 se muestran de color rojo los valores fuera de los umbrales definidos (granularidad: 0-0.2 y de 0.61-1, cohesión: 0.21-1 y complejidad:0-0.9) y en color verde los valores comprendidos en los umbrales definidos (granularidad: 0.21-0.6, cohesión: 0.21-1 y complejidad:0-0.9).

Tabla 36. Valores de las métricas luego de modificar la arquitectura, Transfer-Money

Microservicio	Atributos de Calidad			Métricas		
	Granularidad	Cohesión	Complejidad	SGM	LCOM	NOO
AccountGroupService	0.30	0.40	0.60	0.60	6.00	0.30
AccountService	0.18	0.20	0.90	0.80	9.00	0.18
Customer_AccountWithCustomer Service	0.48	0.30	0.40	0.70	4.00	0.48
CustomerViewService	0.23	0.30	0.70	0.70	7.00	0.23
MoneyTransferService	0.38	0.30	0.50	0.70	5.00	0.38

En la Figura 17 a continuación se muestra un gráfico de líneas que relaciona los valores de los atributos de calidad granularidad, cohesión y complejidad de los microservicios resultantes en la arquitectura modificada. En el gráfico de la Figura 17 el microservicio *Customer_AccountWithCustomerService* es el número 3. Se puede apreciar que el valor del atributo de calidad granularidad del microservicio generado (3) está dentro de los umbrales definidos (0.2-0.6).

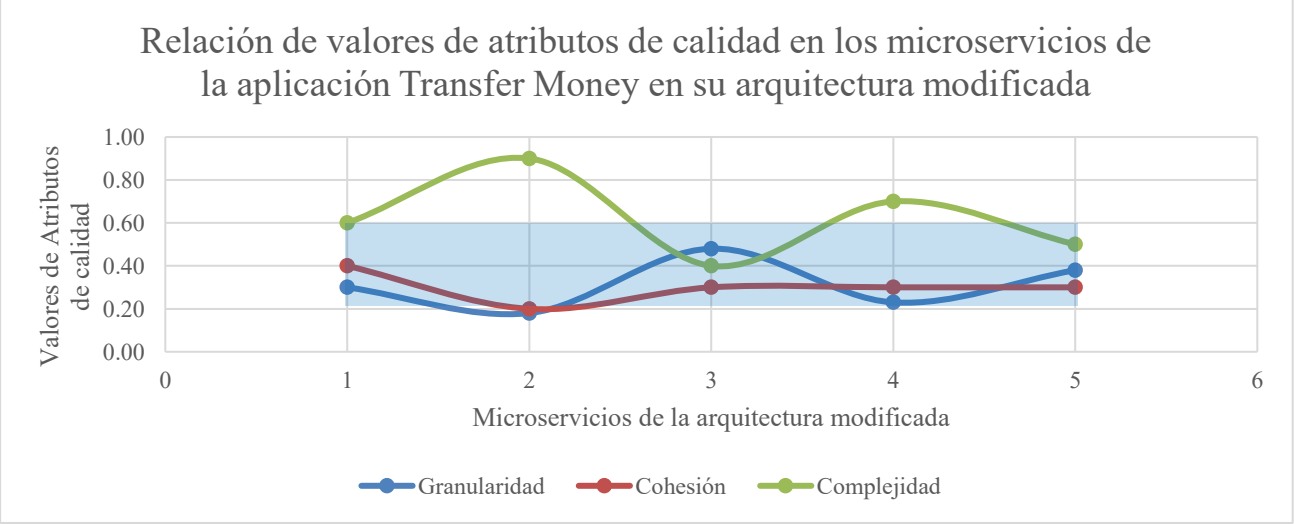

Figura 17. Atributos de calidad luego de la modificación en la arquitectura, Transfer-Money

Para obtener valores de los atributos de calidad de la aplicación completa se procede a determinar el promedio de los valores de las métricas obtenidas por cada uno de los microservicios que la componen. Los valores serán calculados siguiendo las ecuaciones (6), (7) y (8) mostradas en la sección 5.1.7.1

Para una mejor comprensión del comportamiento de los atributos de calidad ante los cambios en la arquitectura se muestran en la Tabla 37 los valores promedio (ASGM, ALCOM y ANOO) de los atributos granularidad, cohesión y complejidad en cada uno de los escenarios (Arquitectura Original y Arquitectura Modificada).

Tabla 37. Promedio de los valores de los atributos de calidad en la arquitectura original y la modificada, Transfer-Money

Arquitectura	Granularidad	Cohesión	Complejidad	Cambios
Original	0.47	0.42	0.52	-
Modificada	0.31	0.30	0.62	Acoplamiento de los microservicios Customer y AccountWithCustomer

En la Figura 18 se representan en una gráfica de barras los valores.

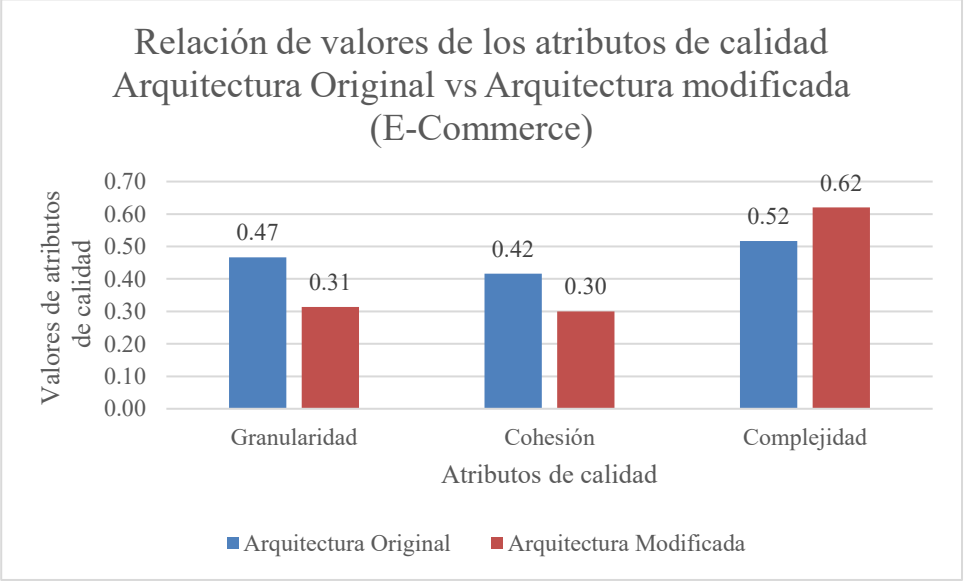

Figura 18. Resumen de valores de los atributos de calidad, Money Transfer.

5.2.2.3. Análisis de CP5 y CP6

Las pruebas "***CP5-*** *Transfer-Money, arquitectura original"* y *"**CP6-**Transfer-Money, arquitectura modificada*" se ejecutan con datos que arrojan las métricas seleccionadas en el modelo de calidad para el balance de atributos de calidad seleccionados en el mismo (ver Figura 4). Estas métricas se obtienen de microservicios que integran la aplicación Transfer-Money.

Para el caso de prueba cinco; que consistió en obtener las métricas de todos los microservicios (6) que conforman la aplicación; luego de recopilar los valores de las métricas y de los atributos de calidad de decidió acoplar dos de los microservicios analizados debido a los valores de granularidad que, en este caso se presentó un microservicio con una sola operación.

Luego de acoplar los microservicios se repite el proceso de obtención de los valores de las métricas. El resumen de los valores obtenidos se muestra en la Figura 18. De este gráfico se puede concluir que, para este caso de estudio (Transfer-Money), al igual que en el caso de estudio E-Commerce, cuando se aumentó la cantidad de operaciones por microservicio el valor de la granularidad disminuyó, la cohesión disminuyó y la complejidad aumentó. En este caso de estudio se puede observar, al igual que en los dos casos de estudio anteriores, que los atributos de calidad cohesión y complejidad se contraponen.

5.3. Conclusiones de las pruebas

Es importante destacar que para los casos de prueba aplicados a los tres casos de estudio presentados se obtuvieron los valores de métricas cuantitativas para código estático, en todos los casos se aplicaron a repositorios obtenidos de Github.

A partir del análisis hecho de los resultados obtenidos se puede concluir que:

- En el caso de los atributos de calidad complejidad y cohesión se ven afectados por la granularidad. Cuando el valor de la granularidad aumenta (menos operaciones por microservicio), la cohesión también lo hace y la complejidad disminuye. En caso contrario, cuando el valor de la granularidad disminuye (más operaciones por microservicio), la cohesión disminuye y la complejidad aumenta.
- Los atributos de calidad cohesión y complejidad son inversamente proporcionales, es decir que se contraponen. En los 6 casos de prueba se observa que cuando los valores de cohesión disminuyen la complejidad aumenta y viceversa.
- En el caso de los tres casos de estudio se observó que cuando se acoplaron microservicios de la aplicación el valor del atributo de calidad granularidad disminuyó. Lo mismo ocurrió en el caso del valor del atributo de calidad cohesión. En el caso del atributo de calidad complejidad, este aumentó. Por otro lado, cuando se dividió un microservicio el valor del atributo de calidad aumentó al igual que el valor del atributo de calidad cohesión. Al contrario, ocurrió con el valor del atributo de calidad complejidad que en este caso disminuyó.

Lo anterior expuesto representa una alternativa para la determinación de la granularidad en microservicios. El modelo de calidad permite conocer cuales atributos se ven afectados por el grado de granularidad de un microservicio. Por otro lado, el balance de los atributos de calidad (cohesión y complejidad) sirve para que el usuario (Arquitecto, Desarrollador de SW, etc.) pueda llevarlo a su contexto y balancearlos según sus necesidades sabiendo el comportamiento de un atributo de calidad con respecto a los otros.

6. Conclusiones y trabajos futuros

En esta sección se presentan las conclusiones de la investigación, los aportes de la investigación al tema en general y las ventajas que brinda el modelo de calidad propuesto a partir de las pruebas realizadas. También se proponen trabajos futuros con el objetivo de profundizar en el tema de la granularidad en microservicios a partir de atributos de calidad.

La conclusión principal de este trabajo es que se logra el objetivo general de esta tesis que fue proponer el modelo de calidad (Figura 4) para determinar la granularidad en microservicios mediante el balance de atributos de calidad.

Los objetivos específicos para el presente trabajo se cumplieron de la siguiente forma:

a) En la literatura consultada se obtuvieron los atributos de calidad que se relacionan con la granularidad. De los documentos, 22 trataron de algún modo el tema de la granularidad. De esos 22, en 9 trabajos se mencionaron atributos de calidad relacionados a la granularidad de microservicios. Finalmente se reconocieron 11 atributos de calidad que se tenían alguna relación con la granularidad en microservicios.

   Para el presente trabajo se seleccionaron 4 atributos de calidad (acoplamiento, cohesión, rendimiento y complejidad) atendiendo a cuánto se repetía en la literatura, a la naturaleza de las métricas propuestas para medirlos y a los umbrales de estas últimas para incluirlos en el modelo de calidad. Para el balance se seleccionaron los atributos cohesión y complejidad.

b) En los trabajos revisados se obtuvo un total de 20 métricas que permiten medir los 4 atributos de calidad definidos en el modelo de calidad incluyendo 9 que se utilizan para calcular el nivel de granularidad de un microservicio (mostradas en el modelo de calidad como métricas directas). De estas 20 métricas se encontraron umbrales o valores de aceptación de 3 de ellas, se incluyeron además 11 que sirven para calcular las 3 que se identificaron con umbrales definidos en la literatura revisada. Las métricas calculadas son Falta de Cohesión (LCOM) para determinar el atributo de calidad Cohesión, Métrica de Granularidad de Servicios (SGM) para determinar el valor del atributo de calidad Granularidad y Número de Operaciones (NOO) para determinar el valor del atributo de calidad Complejidad. Estas métricas se aplican a cada microservicio.

c) En el presente trabajo se evaluaron 3 repositorios de aplicaciones basadas en microservicios con el objetivo de probar el modelo de calidad propuesto de la Figura 4 así como para realizar el balance de los atributos de calidad cohesión y complejidad. Las aplicaciones están desarrolladas en Java utilizando el framework Spring Boot.

   Las pruebas consistieron en medir los atributos de calidad a partir de las métricas en la arquitectura original de la aplicación. Luego de modificar la arquitectura de la aplicación volver a medirlos para determinar el comportamiento de los atributos de calidad en ambos escenarios. Con esto se pudo demostrar la relación existente entre los atributos de calidad propuestos en el modelo de calidad. Además, se realizó el balance entre los atributos de calidad cohesión y complejidad para determinar la granularidad en microservicios.

   Las pruebas realizadas arrojaron que los valores de los atributos de calidad cohesión y complejidad se contraponen, o sea que tienen una relación inversamente proporcional entre ellos. Además, se pudo observar que los valores de cohesión mantienen una relación directamente proporcional con los valores del atributo de calidad granularidad. Por otro lado, los valores de complejidad se comportan de forma inversamente proporcional a los valores del atributo granularidad.

Hoy en día los sistemas de software crecen muy rápidamente. Esto obliga a los desarrolladores a producir un código de calidad. El presente trabajo muestra una guía para el desarrollo de microservicios que sean granulares atendiendo a un balance entre cohesión y complejidad. La investigación proporciona a los arquitectos, analistas y desarrolladores de software una herramienta para tomar decisiones en cuanto al tema de la granularidad óptima de los microservicios. Con el presente trabajo se podrá decidir en cómo balancear los atributos de calidad (en este caso cohesión y complejidad) para obtener una granularidad entre los valores más aceptados según la literatura relacionada con el tema. Es importante destacar que el modelo de calidad y el balance entre atributos es una alternativa más a las ya existentes para el tema de determinación de granularidad en microservicios. Por lo tanto el balance quedará a elección del usuario en dependencia de las exigencias de la arquitectura específica que se trabaje.

6.1. Trabajos futuros

Como trabajos futuros se propone:

- Probar el modelo de calidad en aplicaciones basadas en microservicios que estén desarrolladas en otros lenguajes de programación con el objetivo de probar que el comportamiento del balance de atributos de calidad se mantiene.
- Realizar el balance de atributos de calidad con otros atributos de calidad en tiempo de ejecución como rendimiento y acoplamiento.
- Profundizar en la búsqueda de más métricas que ayuden a medir otros atributos de calidad.
- Automatizar completamente el proceso de obtención de métricas para poder analizar grandes volúmenes de microservicios.
- Probar el atributo de calidad complejidad teniendo en cuenta el número de caminos de decisión independientes de cada operación.

6.2.Publicaciones

Se publicó el artículo “Modelo de calidad para determinar la granularidad de microservicios” en el número 1 del volumen 7 de la revista “Tecnología y ciencia aplicada”. En el mismo se identificaron en la literatura revisada, las métricas y atributos de calidad que se relacionan a la granularidad en microservicios.

Referencias Bibliográficas

[1] S. Li *et al.*, «Understanding and addressing quality attributes of microservices architecture: A Systematic literature review», *Inf. Softw. Technol.*, vol. 131, p. 106449, mar. 2021, doi: 10.1016/j.infsof.2020.106449.

[2] S. Hassan, N. Ali, y R. Bahsoon, «Microservice Ambients: An Architectural Meta-Modelling Approach for Microservice Granularity», abr. 2017. doi: 10.1109/ICSA.2017.32.

[3] N. Kulkarni y V. Dwivedi, «The Role of Service Granularity in a Successful SOA Realization A Case Study», en *2008 IEEE Congress on Services - Part I*, jul. 2008, pp. 423-430. doi: 10.1109/SERVICES-1.2008.86.

[4] S. Hassan, R. Bahsoon, y R. Kazman, «Microservice Transition and its Granularity Problem: A Systematic Mapping Study», 2019, *arXiv*: arXiv:1903.11665. doi: 10.48550/arXiv.1903.11665.

[5] T. de Oliveira Rosa, J. F. L. Daniel, E. M. Guerra, y A. Goldman, «A Method for Architectural Trade-off Analysis Based on Patterns: Evaluating Microservices Structural Attributes», en *Proceedings of the European Conference on Pattern Languages of Programs 2020*, en EuroPLoP '20. New York, NY, USA: Association for Computing Machinery, dic. 2020, pp. 1-8. doi: 10.1145/3424771.3424809.

[6] G. Vale, F. F. Correia, E. Martins Guerra, T. de Oliveira Rosa, J. Fritzsch, y J. Bogner, «Summary of the artifact accompanying the article : "Designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs"», en *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, mar. 2022, pp. 57-57. doi: 10.1109/ICSA-C54293.2022.00020.

[7] J. Lewis y M. Fowler, «Microservices: a definition of this new architectural term», martinfowler.com. Accedido: 2 de marzo de 2023. [En línea]. Disponible en: https://martinfowler.com/articles/microservices.html

[8] F. H. Vera Rivera, «Modelo inteligente de especificación de la granularidad de aplicaciones basadas en microservicios.», Universidad del Valle, Cali, Colombia, 2021. Accedido: 19 de febrero de 2023. [En línea]. Disponible en: https://bibliotecadigital.univalle.edu.co/handle/10893/19567

[9] D. A. Barrios Contreras, «Arquitectura de Microservicios | Tecnología Investigación y Academia», vol. 6, n.º 1, abr. 2018, Accedido: 27 de febrero de 2023. [En línea]. Disponible en: https://revistas.udistrital.edu.co/index.php/tia/article/view/9687

[10] E. Albertos Gómez, «Arquitecturas software para microservicios: una revisión sistemática de la literatura», masters, E.T.S.I de Sistemas Informáticos (UPM), 2018. Accedido: 27 de febrero de 2023. [En línea]. Disponible en: https://oa.upm.es/51460/

[11] «ISO 25010». Accedido: 7 de agosto de 2023. [En línea]. Disponible en: https://iso25000.com/index.php/normas-iso-25000/iso-25010

[12] F. Driessen, «A quantitative assessment method for microservices granularity to improve maintainability».

[13] *IEEE Standard for a Software Quality Metrics Methodology*, marzo de 1993. doi: 10.1109/IEEESTD.1993.115124.

[14] J. Bermeo Conto, M. Sánchez, J. MaldonadoJ., y J. P. Carvallo, «Modelos de Calidad de Software en la Práctica: Mejorando su Construcción con el Soporte de Modelos Conceptuales», 2016.

[15] O. Al-Debagy y P. Martinek, «A Metrics Framework for Evaluating Microservices Architecture Designs», *J. Web Eng.*, vol. 19, n.º 3–4, pp. 341-370, jun. 2020, doi: 10.13052/jwe1540-9589.19341.

[16] D. Shadija, M. Rezai, y R. Hill, «Microservices: Granularity vs. Performance», en *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, Austin Texas USA: ACM, dic. 2017, pp. 215-220. doi: 10.1145/3147234.3148093.

[17] J.-P. Gouigoux y D. Tamzalit, «From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture», presentado en IEEE International Conference on Software Architecture Workshops, 2017, pp. 62-65. doi: 10.1109/ICSAW.2017.35.

[18] A. Homay, A. Zoitl, M. de Sousa, M. Wollschlaeger, y C. Chrysoulas, «Granularity Cost Analysis for Function Block as a Service», jul. 2019, p. 1204. doi: 10.1109/INDIN41052.2019.8972205.

[19] C. Merino Ibanez, «Metodología en la determinación de la granularidad de un Microservicio», CENIDET, Cuernavaca, México, 2023. Accedido: 27 de febrero de 2023. [En línea]. Disponible en: https://rinacional.tecnm.mx/jspui/handle/TecNM/4961

[20] O. Al-Debagy y P. Martinek, «Extracting Microservices' Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach», en *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*, jun. 2020, pp. 289-294. doi: 10.1109/SoSE50414.2020.9130466.

[21] J. Bogner, S. Wagner, y A. Zimmermann, «Collecting Service-Based Maintainability Metrics from RESTful API Descriptions: Static Analysis and Threshold Derivation», vol. 1269, 2020, pp. 215-227. doi: 10.1007/978-3-030-59155-7_16.

[22] S. Klock, J. M. E. M. van der Werf, J. P. Guelen, y S. Jansen, «Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures», en *2017 IEEE International Conference on Software Architecture (ICSA)*, abr. 2017, pp. 11-20. doi: 10.1109/ICSA.2017.38.

[23] M.-D. Cojocaru, A. Uta, y A.-M. Oprescu, «Attributes Assessing the Quality of Microservices Automatically Decomposed from Monolithic Applications», en *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, Amsterdam, Netherlands: IEEE, jun. 2019, pp. 84-93. doi: 10.1109/ISPDC.2019.00021.

[24] J. de la Cruz y I. Daniel, «Selección y adaptación de métricas para Microservicios», abr. 2022, Accedido: 27 de febrero de 2023. [En línea]. Disponible en: https://rinacional.tecnm.mx/jspui/handle/TecNM/4422

[25] M. Cojocaru, A. Uta, y A.-M. Oprescu, «MicroValid: A Validation Framework for Automatically Decomposed Microservices», en *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, dic. 2019, pp. 78-86. doi: 10.1109/CloudCom.2019.00023.

[26] I. S. Mihai, «A Systematic Evaluation of Microservice Architectures Resulting from Domain-Driven and Dataflow-Driven Decomposition», 2023.

[27] L. Baresi y M. Garriga, «Microservices: The Evolution and Extinction of Web Services?»

[28] M. Mazzara, A. Bucchiarone, N. Dragoni, y V. Rivera, «Size Matters: Microservices Research ans Applications», Springer, pp. 29-42. [En línea]. Disponible en: https://doi.org/10.1007/978-3-030-31646-4

[29] F. H. Vera-Rivera, C. Gaona, y H. Astudillo, «Defining and measuring microservice granularity—a literature overview», *PeerJ Comput. Sci.*, vol. 7, p. e695, sep. 2021, doi: 10.7717/peerj-cs.695.

[30] H. Vural, M. Koyuncu, y S. Misra, «A Case Study on Measuring the Size of Microservices», jul. 2018, pp. 454-463. doi: 10.1007/978-3-319-95174-4_36.

[31] T. Blokehead, *Cómo construir Microservicios : Los diez principales trucos para modelar, integrar y desplegar microservicios*. Babelcube Inc., 2017.

[32] Amin Espinoza, *¿Qué son los microservicios? ¿Y por qué tanto hype?*, (23 de junio de 2022). Accedido: 12 de septiembre de 2023. [En línea Video]. Disponible en: https://www.youtube.com/watch?v=f7k6WuwIh8k

[33] M. Imranur, Rahman, S. Panichella, y D. Taibi, «A curated Dataset of Microservices-Based Systems», 7 de septiembre de 2019, *arXiv*: arXiv:1909.03249. doi: 10.48550/arXiv.1909.03249.

[34] D. Taibi y K. Systä, «A Decomposition and Metric-Based Evaluation Framework for Microservices», en *Cloud Computing and Services Science*, 2020, pp. 133-149. Accedido: 18 de septiembre de 2023. [En línea]. Disponible en: https://www.springer.com/series/7899

[35] F. Haupt, F. Leymann, A. Scherer, y K. Vukojevic-Haupt, «A Framework for the Structural Analysis of REST APIs», en *2017 IEEE International Conference on Software Architecture (ICSA)*, abr. 2017, pp. 55-58. doi: 10.1109/ICSA.2017.40.

[36] O. Al-Debagy y P. Martinek, «A New Decomposition Method for Designing Microservices», *Period. Polytech. Electr. Eng. Comput. Sci.*, vol. 63, jun. 2019, doi: 10.3311/PPee.13925.

[37] J. Rashid, T. Mahmood, y M. W. Nisar, «A Study on Software Metrics and its Impact on Software Quality», vol. 24, n.º 1.

[38] Y. León Perdomo, A. Enrique Góngora Rodríguez, y A. Febles Estrada, «Aplicando métricas de calidad a proyectos y procesos durante las pruebas exploratorias», *Rev. Cuba. Cienc. Informáticas*, vol. 7, n.º 2, pp. 193-205, jun. 2013.

[39] J. Bogner, S. Wagner, y A. Zimmermann, «Automatically measuring the maintainability of service- and microservice-based systems: a literature review», en *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, Gothenburg Sweden: ACM, oct. 2017, pp. 107-115. doi: 10.1145/3143434.3143443.

[40] S. Aizprua, A. Ortega, y L. V. Chong, «Calidad del software una perspectiva continua», *Rev. Científica Cent.*, vol. 8, n.º 2, Art. n.º 2, 2019.

[41] A. M. Ocaña, «Comunicación entre Microservicios: ¿Compartimos código?», Adictos al trabajo. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://www.adictosaltrabajo.com/2017/01/12/comunicacion-entre-microservicios-compartimos-codigo/

[42] F. H. Vera-Rivera, H. Astudillo, y C. Gaona, «Desarrollo de aplicaciones basadas en microservicios: tendencias y desafíos de investigación», *RISTI - Rev. Ibérica Sist. E Tecnol. Informação*, n.º E23(2019), pp. 107-120, oct. 2019.

[43] I. Jiménez y V. Alberto, «Desarrollo de software basado en microservicios: un caso de estudio para evaluar sus ventajas e inconvenientes», Proyecto/Trabajo fin de carrera/grado, Universitat Politècnica de València, 2018. Accedido: 27 de febrero de 2023. [En línea]. Disponible en: https://riunet.upv.es/handle/10251/111173

[44] T. Černý, M. Donahoo, y J. Pechanec, «Disambiguation and Comparison of SOA, Microservices and Self-Contained Systems», sep. 2017, pp. 228-235. doi: 10.1145/3129676.3129682.

[45] J. L. Cueva Tacuri, «Identificación y aplicación de un modelo para evaluar la interoperabilidad en Microservicios», bachelorThesis, 2017. Accedido: 27 de febrero de 2023. [En línea]. Disponible en: http://dspace.utpl.edu.ec/jspui/handle/20.500.11962/20692

[46] H. Liu *et al.*, «JCallGraph: Tracing Microservices in Very Large Scale Container Cloud Platforms», en *Cloud Computing – CLOUD 2019*, D. Da Silva, Q. Wang, y L.-J. Zhang, Eds., Cham: Springer International Publishing, 2019, pp. 287-302.

[47] C. B. Pareja Valerio y L. J. Burgos Robles, «La arquitectura de software basada en microservicios: Una revisión sistemática de la literatura», Universidad Peruana Unión, 2019. Accedido: 2 de marzo de 2023. [En línea]. Disponible en: https://repositorio.upeu.edu.pe/handle/20.500.12840/2524

[48] C. E. Gutiérrez Herrera, «Metodología para calcular métricas de software a partir del catálogo de requisitos empelando el análisis de compensación», *Univ. Católica St. María*, mar. 2022, Accedido: 10 de marzo de 2023. [En línea]. Disponible en: https://repositorio.ucsm.edu.pe/handle/20.500.12920/11613

[49] O. Al-Debagy, «Microservices Identification Methods and Quality Metrics», Budapest University of Technology and Economics, Hungría, 2021.

[50] Á. I. Rodríguez, J. I. Padilla, y H. A. Parra, «Vista de Arquitectura basada en microservicios para aplicaciones web», vol. 7, n.º 1, 2019. Accedido: 18 de febrero de 2023. [En línea]. Disponible en: https://revistas.udistrital.edu.co/index.php/tia/article/view/13364/15929

[51] O. Al-Debagy y P. Martinek, «A Metrics Framework for Evaluating Microservices Architecture Designs», *J. Web Eng.*, vol. 19, n.º 3–4, pp. 341-370, jun. 2020, doi: 10.13052/jwe1540-9589.19341.

[52] J. Bogner, S. Wagner, y A. Zimmermann, *Collecting Service-Based Maintainability Metrics from RESTful API Descriptions: Static Analysis and Threshold Derivation*. 2020. doi: 10.1007/978-3-030-59155-7_16.

[53] J. Grohmann, *Train Ticket : A Benchmark Microservice System*. Java. [En línea]. Disponible en: https://github.com/jo102tz/train-ticket

[54] C. Richardson, *Customers, accounts and money transfers*. [En línea]. Disponible en: https://github.com/crctraining/customers-accounts-and-money-transfers

[55] *REST Microservices architecture for E-commerce*. [En línea]. Disponible en: https://github.com/RainbowForest/e-commerce-microservices?tab=readme-ov-file

Anexos

Anexo 1. Trabajos relacionados con análisis de la granularidad en tiempo real

En [2] se presenta Ambient-PRISMA Textual Language como un potencial lenguaje para definición de microservicios, concebido para realizar el cambio dinámico de la granularidad de los microservicios en tiempo de ejecución. Asimismo, se realiza una asignación dinámica de la granularidad de una aplicación basada en microservicios en tiempo de ejecución. La herramienta llamada Microservice Ambients establece las primitivas para adaptar los límites o boundaries. En este trabajo no se hace un análisis de la calidad de las divisiones hechas ni tienen en cuenta atributos de calidad.

En [16], se hace mención del potencial de la granularidad para influir en la latencia de las aplicaciones. En este trabajo [16], se compara el comportamiento del rendimiento de una aplicación basada en microservicios en cuatro escenarios posibles utilizando la herramienta Dynatrace (https://www.dynatrace.com). Como conclusiones de este trabajo los autores exponen que la latencia de una aplicación aumenta y como consecuencia disminuye el rendimiento al ser más granulares los microservicios que la componen. En este estudio se relaciona el atributo de calidad rendimiento con la granularidad, sin embargo, no aparecen definidas las métricas para medirlo, solamente se hacen pruebas en diferentes escenarios y se hacen comparaciones. Las pruebas en esta investigación se realizaron en tiempo de ejecución.

En [17], los autores determinan la granularidad en tiempo de ejecución haciendo un balance entre las características costo del aseguramiento de la calidad y costo de despliegue. Tampoco aquí se analizan atributos de calidad ni métricas relacionadas. Además, es un trabajo basado en prueba y error de experiencias técnicas realizadas a una aplicación para descomponerla en microservicios.

Por otro lado en [18] se hace un análisis de granularidad para Function Block (Bloque de Funciones) como servicios. Para ello se sugiere utilizar un enfoque de cálculo de complejidad y gastos generales llamado Análisis de costos de granularidad del servicio (SGCA). La SGCA introduce dos funciones básicas, el Análisis de Costos de Descomposición de Servicios (SDCA) y el Análisis de Costos de Composición de Servicios (SCCA), que consideran el consumo de tiempo como un costo para

penalizar los métodos de (des)composición de servicios utilizados. En este trabajo no se especifican métricas.

Anexo 2. Trabajos relacionados con análisis de la granularidad en tiempo de diseño

En el trabajo [8] Vera Rivera propone un modelo matemático para determinar la granularidad en microservicios en tiempo de diseño, basándose en historias de usuario. En este trabajo se definen atributos de calidad (complejidad, acoplamiento, cohesión) y se utilizan métricas obtenidas de la literatura consultada para la medición de estos atributos. El autor define la granularidad en microservicios como un vector compuesto por los valores obtenidos al aplicar las métricas adoptadas para cada uno de los atributos de calidad. En el trabajo no se especifican umbrales para las métricas que utilizan en el modelo, solamente hacen comparaciones de la granularidad entre aplicaciones basadas en microservicios. En este mismo orden de ideas en [19] el autor propone una metodología para determinar la granularidad en tiempo de diseño a partir de la especificación de casos de usos de la aplicación. En este trabajo solo se analiza el diseño de la aplicación por lo que no se hace un análisis de métricas al no necesitar analizar código fuente.

Anexo 3. Mapeos sistemáticos de la literatura

Hassan et al. en [4] hace un mapeo sistemático de la literatura evaluando la transición de un monolito a una arquitectura de microservicios, aquí el autor define el término microservilización refiriéndose a la antes mencionada transición. Se plantea que uno de los problemas fundamentales de la transición a los microservicios es analizar su nivel de granularidad. En este trabajo no se especifican métricas para medir los atributos de calidad expuestos.

Anexo 4. Métricas para cohesión

Para el caso del atributo de calidad Cohesión se encontraron las siguientes métricas:

- ✓ **LCOM** (Falta de cohesión) que según [15] mide la cohesión o, en otras palabras, la similitud entre operaciones en un servicio específico y si esas operaciones están relacionadas entre ellas. En [15], la métrica LCOM es basada en la métrica de falta de cohesión para POO de Henderson-Sellers. Esta consiste en encontrar cuantas veces ha sido usado un parámetro específico en un determinado microservicio dividiendo el producto del número de operaciones multiplicado por el número de parámetros únicos. En el estudio hecho en este trabajo se define como umbral de esta métrica valores entre 0 y 0.8.

- ✓ **SIDC** (Cohesión de datos de Interface de Servicio) utilizada en [12] se describe que es otra noción de cohesión, que es una métrica más específica del servicio, ya que se mide a nivel de interfaz. La SIDC considera la igualdad entre los tipos de parámetros de las operaciones en una interfaz, de modo que si todas las operaciones definidas en una interfaz utilizan un tipo de datos de parámetro común, esto indica que el servicio correspondiente es altamente cohesivo. Se calcula a partir de la suma de operaciones con tipos de datos idénticos dividida por el número total de operaciones distintas. Los valores de en aceptación o umbrales definidos para esta métrica son 0.55 a 1.

Anexo 5. Métricas para acoplamiento

- ✓ Según el autor en [8] $\boldsymbol{AIS}$ es el número de otros microservicios que invocan al menos una operación de la interfaz de un microservicio. AIS_i es el número de clientes (otros microservicios) que invocan al menos una operación de ms_i.
- ✓ También en [8] el autor también refiere $\boldsymbol{ADS}$ es el número de otros microservicios de los que depende el microservicio. ADS_i es el número de microservicios de los cuales se invoca al menos una operación de ms_i
- ✓ El mismo autor refiere en [8] que $\boldsymbol{SIY}$ número de pares de microservicios interdependientes. En este caso SIY_i define el número de pares de microservicios que dependen bidireccionalmente uno del otro dividido por el número total de microservicios