UNIVERSIDAD DEL NORTE

DOCTORADO EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS

Búsqueda y Selección de Servicios Web con restricciones QoS en ambientes Cloud Computing Ar_WSDS

Marco Antonio Adarme Jaimes

Barranquilla, 2021

UNIVERSIDAD DEL NORTE

TESIS

Búsqueda y Selección de Servicios Web con restricciones QoS en ambientes Cloud Computing
Ar_WSDS

Autor: Marco Antonio Adarme Jaimes

Tutor: Dr. Miguel Angel Jimeno Paba

Barranquilla, 2021

Por qué tu tiempo es perfecto, gracias Padre Celestial.
A mi Músico en el cielo, a mi Reina Madre, Hermana y a mi Princesa
su motivación y apoyo fueron la fuerza para culminar.

Resumen

En la actualidad la nube enfrenta problemas para la búsqueda y selección de servicios web, debido al gran número de servicios disponibles por los diferentes proveedores cloud y a la diversidad de criterios de calidad de servicio (QoS, por sus siglas en inglés) de cada uno de ellos. Muchos servicios pueden satisfacer un requisito con criterios de calidad similares. Debido a esto, los desarrolladores de software tienen que elegir los servicios más adecuados para una composición determinada. Esta tesis desarrolla una estrategia para el descubrimiento y selección de servicios web denominada Ar_WSDS, que basa su implementación en el funcionamiento sistemático de reconocimiento de patrones del cerebro. Los servicios web son representados como patrones hacer reconocidos por Ar_WSDS, quien determina los servicios necesarios y suficientes que constituyen la composición y que cumplan con los criterios QoS que el desarrollador defina. El proceso de reconocimiento da como resultado un conjunto de servicios web que cumplen con una métrica de calidad para cada uno de sus servicios como para el servicio compuesto.

Ar_WSDS ofrece una nueva visión de reconocimiento que permite dividir el problema de selección en tareas funcionales y descriptivas. El sistema es concebido desde una arquitectura por componentes, principio que brinda granularidad y adaptación para la adición nuevos criterios QoS. Los resultados de la evaluación del sistema demuestran la efectividad del proceso de descubrimiento y selección, y de categorización o ranking de servicios que ofrecen un mayor rango de búsqueda y selección, frente a otros enfoques.

Abstract

Currently, Cloud is facing problems in the search and selection of web services due to many services available from different cloud providers and the diversity of QoS (Quality of Service) criteria of each service. Multiple services can satisfy a requirement with similar QoS criteria. Due to this, software developers have to choose the most appropriate services for a given composition. This thesis develops a strategy for web service discovery and selection called Ar_WSDS, which bases its implementation on the systematic pattern recognition performance of the brain. Web services are represented as patterns to be recognized by Ar_WSDS, which determines the necessary and sufficient services that constitute the composition and meet the developer's QoS criteria. The recognition process results in a set of web services that meet a quality metric for each of their services and the composite service.

Ar_WSDS shows a new vision of recognition that allows the selection problem to be divided into functional and descriptive tasks. The system is conceived as a component-based architecture, a principle that provides granularity and adaptability for new QoS criteria. The system's evaluation results show the effectiveness of the discovery and selection process and the categorization or ranking of services that offer a more comprehensive range of search and selection versus other approaches.

Agradecimientos

A mi *Universidad Francisco de Paula Santander(Cúcuta, Col)*, por su apoyo económico y académico en la realización de la investigación.

A la *Universidad del Norte(Barranquilla, Col)*, por su equipo de docentes que aportaron su experiencia y conocimientos en mi formación como investigador.

A mi Tutor, *Dr. Miguel Jimeno*, por su constante motivación, quién creyó y apoyo cada fase de la realización de esté proyecto.

Marco Antonio Adarme Jaimes
Barranquilla, Colombia, 2021

Notación y acrónimos

NOTACIÓN

$\oplus$	:	OR exclusivo (XOR)
WS_i	:	Servicio web atómico dentro del modelo de composición
WS_T	:	Conjunto de Servicios Web del modelo de composición
QoS_T	:	Conjunto de arreglos de criterios de calidad de los servicios web del modelo de composición
WS_g	:	Conjunto de servicios web a ser reconocidos por el sistema
WS_c	:	Conjunto de servicios web candidatos de un modelo de composición
Γ	:	Módulo de reconocimiento
ρ	:	Patrón
$M()$	:	Función matching de reconocimiento
φ	:	Valor del ranking de un servicio web
φ'	:	Valor del ranking de un QoS para un servicio web
Δ	:	Umbrales de calidad
β	:	Función de penalidad de calidad

LISTA DE ACRÓNIMOS

ABB	:	Árbol binario de búsqueda
ACM	:	Modelo Abstracto de Composición
CCOA	:	Cloud Computing Open Architecture
HTTP	:	Hypertext Transfer Protocol
IaaS	:	Infraestructura como servicio
IT	:	Tecnología de la Información
JSON	:	JavaScript Object Notation
KS	:	Señales claves
PS	:	Señales parciales
QoS	:	Calidad de Servicio
REST	:	Representational State Transfer
RNF	:	Requerimiento no funcional
RF	:	Requerimiento funcional
SaaS	:	Software como un servicio
SOAP	:	Simple Object Access Protocol
SP	:	Rutas de solución
SOA	:	Arquitectura orientada a servicios
UDDI	:	Description Discovery and Integration
UML	:	Unified Modeling Language
WS	:	Servicio Web
WSC	:	Composición de Servicio Web
WSDL	:	Web Services Description Language
WSCI	:	Web Service Choreography Interface
XML	:	Extensible Markup Language

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

CAPÍTULO 1
Introducción

La composición de servicios web (WSC) se define como la combinación lógica de servicios web (WS) que ofrece un nivel de colaboración e intercambio de datos entre ellos de forma sincronizada para satisfacer un requerimiento funcional (RF). El servicio constituye un componente que expone sus funcionalidades a través de interfaces bajo un lenguaje específico de definición[1, 2]. Estas funcionalidades son las que en combinación con otros servicios satisfacen el requerimiento. En la actualidad la tarea de composición no sólo depende de colocar servicios lógicamente dispuestos, también intervienen aspectos definidos en sus requerimientos no funcionales (RNF) que constituyen criterios de calidad de servicio(QoS) para seleccionar los WS más adecuados para una composición. [3].

Los desarrolladores se ven enfrentados a seleccionar de un grupo de servicios aquellos que pueden intervenir en la composición que fue diseñada. La selección de un servicio a partir de las preferencias del desarrollador presentan inconvenientes definidos en sus RNF, principalmente en mantener el control y vigilancia del funcionamiento del servicio durante la ejecución de la composición[4]. De está forma , cualquier sistema que incorpore la noción de composición para el desarrollo de sus aplicaciones deben implementar sistemas de búsqueda y selección de servicios que aseguren su correcto funcionamiento. Está tarea, por lo general se realiza de manera manual con la participación directa del desarrollador y cuyos criterios de selección se basan únicamente de los servicios que puede analizar de forma rápida y en los repositorios que tenga predefinidos.

Con el advenimiento de las tecnologías basadas en cloud computing, el despliegue de los servicios se realiza bajo proveedores que gestionan sus recursos con alto niveles de detalle, de esta forma cualquier WS será supervisado para mantener el control de lo que exponen y como sus clientes irán a consumirlos, paralelo a los QoS que puede variar de un proveedor a otro.

1.1. PLANTEAMIENTO DEL PROBLEMA

En una actividad de composición cada servicio intercambiará mensajes con sus pares a través de conectores de un servicio atómico (un sólo WS) y a partir de él se desarrollará un servicio

complejo o compuesto(unión de varios WS) que interactuará a través de alguna lógica funcional. Su nivel de escalabilidad y crecimiento dependerá en gran medida de la flexibilidad de los servicios seleccionados, de tal forma que si un servicio es independiente con criterios QoS constantes estos podrán ser usados posteriormente por otro servicio compuesto[4].

Desde la perspectiva más general, el problema de descubrir y la seleccionar servicios web se enfrentan actualmente a problemas de optimización, muchos servicios pueden satisfacer un requisito con criterios QoS similares. Por ello, los desarrolladores de software tienen que elegir los servicios más adecuados para una composición determinada, tarea compleja debido al rápido aumento de proveedores y servicios disponibles en la cloud.

Con el crecimiento exponencial de servicios con diferentes niveles QoS [5] e igual funcionalidad, hacen que la tarea de seleccionar un WS por parte de los desarrolladores de software no sea trivial, máxime si existen requerimientos no funcionales que deben tenerse en cuenta y que deben soportar ambientes cloud con parámetros de calidad (QoS) que aseguran el cumplimiento tanto de las necesidades de los solicitantes como la de los proveedores y la capacidad de establecer acuerdos de disposición y adquisición de recursos computacionales(negociaciones de forma estática o dinámica) si así son requeridos[6].

Los entornos cloud conllevan a problemas intrínsecos generados por los proveedores cuyo despliegue de servicios en diferentes nodos[7], pueden generar una variación de criterios QoS; por lo tanto, los requerimientos no funcionales deben ser analizados con mucha precaución a la hora de descubrir y seleccionar servicios.

Del análisis precedente, se formulan las siguientes preguntas de investigación:

1. ¿Cuáles son los problemas específicos que enfrenta una composición de servicios en su etapa de búsqueda y selección de WS cuando operan con restricciones QoS?
2. ¿Cuál sería la estrategia de solución para buscar y seleccionar WS bajo condiciones QoS que satisfaga los requerimientos no funcionales de una composición de servicios?
3. ¿Qué análisis se tiene que llevar a cabo a nivel de QoS para categorizar un WS como el óptimo para una composición de servicios?

Para abordar este problema, esté trabajo crea una estrategia computacional denominada «Algoritmo de Reconocimiento de Patrones para el Descubrimiento y Selección de Servicios Web » por sus siglas **Ar_WSDS** usando el funcionamiento sistemático del cerebro humano para sus procesos de clasificación y categorización. Los WS son representados como patrones para ser reconocidos por Ar_WSDS, quien determina los servicios necesarios y suficientes que constituyen la composición de servicios que cumpla con sus requerimientos funcionales y no funcionales (QoS). Este proceso permite clasificar los servicios mediante módulos de reconocimiento creados dinámicamente en función de sus parámetros de calidad, dando como resultado un conjunto de servicios web para una composición en particular. Este enfoque permite resolver el problema de selección mediante tareas menos complejas. Ar_WSDS, está orientado a trabajar con parámetros QoS a nivel atómico y del servicio complejo. De esta forma, es posible tener personalización de criterios QoS diferentes para cada uno de sus servicios y establecer un métrica de calidad global que estos deben alcanzar.

La noción del reconocimiento de patrones ofrece métodos y algoritmos para la clasificación de objetos basado en el análisis de sus atributos. El concepto de clasificación modela el sistema en entidades de información capaces de resolver las incógnitas en los patrones y permitiendo eliminar el ruido en la información para que su tratamiento sea más eficiente. En el caso particular del ecosistema de servicios su información se encuentra en repositorios que son alimentados por los diferentes proveedores cada cierto periodo de actualización. Esta información generalmente se encuentra incompleta y el análisis de las valores de los criterios QoS no son exactos, la mayoría de las investigaciones en está área se basan de datasets que ofrecen la estructuración de los repositorios de una manera más estandarizada.

Ar_WSDS formaliza la idea de composición de servicios y su actividad de selección a través de un modelo lógico-matemático de mecanismos de reconocimiento de WSs en dos fases, la primera que describe la búsqueda sintáctica del servicio y la segunda que ofrece la selección y clasificación a través de parámetros QoS basados en dos estrategias de reconocimiento, denominadas por señales claves y parciales, respectivamente. Cada QoS es analizado según su naturaleza (tipo parámetro y valores máximos o mínimos), de esta forma, el reconocimiento por señales claves cada servicio atómico será reconocido en umbrales cerrados, la segunda estrategia por señales parciales, los criterios son analizados a través de niveles de aceptación a partir de su valor promedio.

El clasificador permite tener un mecanismo para la jerarquización o ranking de servicios que calcula la puntuación del servicio en función de los parámetros QoS. Esté mecanismo de ranking ofrece una gama de soluciones en donde el desarrollador podrá escoger los servicios más adecuados según la calidad que desee obtener para su composición. El modelo final se implementa de manera adaptativa lo que permite que sus módulos de reconocimiento puedan ser provistos de nuevos parámetros QoS.

Ar_WSDS propone para los desarrolladores una forma novedosa de seleccionar servicios a partir de métricas definidas por criterios QoS ofreciendo un apoyo en la etapa del diseño de la composición donde se deben establecer los servicios involucrados en ella y como estos afectan a la aplicación que los irá a desplegar, con la estrategia diseñada el desarrollador podrá analizar el comportamiento a nivel molecular y atómico de sus servicios. Adicional a esto, el mecanismo puede ser utilizado en otros contextos donde el concepto de criterio QoS sea utilizado, a citar algunos:

- Selección de microservicios a partir de criterios de calidad de una arquitectura dada.
- Jerarquización de arquitecturas cloud a partir de criterios de calidad de proveedores.
- Sistemas orquestadores para seleccionar el funcionamiento y disponibilidad de elementos software, para proveer funciones clasificadoras que un contenedor debe tener bajo unas métricas de utilización.

A continuación se describen los objetivos, los resultados de investigación obtenidos y aportes académicos en el desarrollo de la tesis.

1.2. DESCRIPCIÓN Y DESARROLLO DE OBJETIVOS

La tesis desarrolla un mecanismo para la selección y búsqueda de WS para proceso de composición, basados en cloud bajo parámetros QoS preestablecidos así como aquellos que surjan durante la actividad de composición que afecte directamente los requerimientos no funcionales del servicios complejo que se desea ejecutar. Para lograr este fin se realizaron los siguientes objetivos descritos en la Tabla 1.1.

Tabla 1.1: Descripción de objetivos y Resultados

Objetivo	**Resultados**
General: Proponer un mecanismo para la búsqueda y selección de servicios web con restricciones QoS para procesos de composición en cloud computing	Creación de la estrategia computacional Ar_WSDS.
Específico: Caracterizar las diferentes aplicaciones para la selección y búsqueda orientada a WS en ambientes SOA convencionales y sobre Cloud Computing.	Elaboración de estado de arte de mecanismos de selección en entornos cloud y con procesos de búsqueda usando restricciones QoS. El estado de arte se realizó a través de una propuesta de análisis documental basados en la teoría de la moda.
Específico: Diseñar una arquitectura para el componente de selección y búsqueda de WS que ofrezca la posibilidad de ser configurable dinámicamente a parámetros QoS.	Creación de estrategia computacional que analiza WS atómicos y moleculares. Su proceso de búsqueda y selección es configurable para cual modelo de composición. En este punto,se Realizó un modelo de especificación de datos para identificar proveedores y WS en cloud.
Específico: Diseñar una estrategia para la categorización de servicios web.	Creación de la estrategia para ranking de servicios proporcionando análisis de cada requerimiento no funcional a nivel atómico y de la calidad del servicio compuesto.

Continúa en la página siguiente.

Objetivo	Resultados
Específico: Elaborar casos de prueba vinculados al proceso de búsqueda y selección de servicios web.	Se desarrolla la fase experimental basados en el proyecto QWS Dataset [8],así mismo se compara la estrategia con dos enfoques de selección de servicios. Los experimentos son llevados a cabo en una aplicación con un backend realizado en phyton y dos frontends, el primero en modo consola para búsquedas masivas de servicios y el segundo un vista web a través del sitio: http://arwsds.madarme.co/, donde se presenta el proceso de simulación del sistema.

1.3. RESULTADOS Y APORTES DE INVESTIGACIÓN

La investigación presenta los siguientes aportes académicos y científicos en el área de cloud computing, servicios web, composición de servicios y reconocimiento de patrones:

1. Un modelo teórico para el reconocimiento de patrones en el área de composición de servicios para procesos de descubrimiento y selección de WS mediante procesos de segmentación de variables basado en el análisis de QoS. Este proceso ofrece una forma simple de descomponer el problema de naturaleza combinatoria en una estrategia de búsqueda y selección de servicios basado en módulos de reconocimiento dinámico.
2. La formalización lógica matemática de servicios web a nivel atómico y molecular de una composición, a través de la representación de entidades de información que representan patrones y señales.
3. Una estrategia computacional de jerarquización (ranking) que calcula la puntuación del servicio en función de los parámetros de QoS establecidos por el desarrollador de software. Este mecanismo de ranking ofrece una gama de soluciones donde el desarrollador podrá escoger los servicios más adecuados según sus requerimientos no funcionales que tenga definido para su composición.
4. Una implementación de la estrategia computacional de selección y jerarquización creada en está investigación, basado en una arquitectura modular dinámica a partir de nueve criterios QoS (ver Tabla 5.1). El diseño de la implementación facilita su escalabilidad para la adición de nuevos parámetros de QoS sin necesidad de alterar su programación base.
5. Una propuesta de concepto de reconocimiento de patrones de componentes de software. Esta nueva idea de reconocimiento tratará estrategias computacionales encaminadas al reconocimiento de atributos en unidades de código de programación, a nivel de las métricas de evaluación que se tengan para cada tipo de componente de software.

Los aportes de la investigación fueron socializados en los siguientes artículos en revistas especializadas y en actividades de apropiación social de conocimiento y divulgación científica:

- Artículos:
  1. M. Adarme and M. Jimeno, "QoS-Based Pattern Recognition Approach for Web Service Discovery: Ar_WSDS," Applied Sciences, vol. 11, no. 17, p. 8092, Aug. 2021. *(ISI-Q2 Scopus-Categoría A1, Publindex).*
  2. M. Adarme, M. Jimeno, and E. G. Puerto, "Web services selection a perspective of computational physics," J. Phys. Conf. Ser., vol. 1587, p. 012017, Sep. 2020, doi: 10.1088/1742-6596/1587/1/012017. *(Q4 Scopus-Categoría B, Publindex).*
  3. M. Adarme, M. Jimeno, and E. G. Puerto, "A recursive pattern recognition approach to selection web services in cloud environment," J. Phys. Conf. Ser., vol. 1513, p. 12004, Mar. 2020, doi: 10.1088/1742-6596/1513/1/012004. *(Q4 Scopus-Categoría B, Publindex).*
- Eventos científicos:
  1. VI Congress on Innovation and Appropriation of the Technologies of Information and Communications (VI CIATIC) (2018). "A recursive pattern recognition approach to selection web services in cloud environment".
  2. VI International Week of Science, Technology and Innovation(2019). "Selecting and searching mechanisms of web services with QoS conditions in cloud computing".
  3. 8th International Week of Science, Technology and Innovation(2021). "QoS-based web service Ranking using pattern recognition".

1.4. ESTRUCTURA DEL DOCUMENTO

La investigación se estructura en seis capítulos. El esquema se organiza de modo que se parte de los aspectos más genéricos y se van concretando el concepto de reconocimiento de patrones en el área de composición de servicios web en su proceso de descubrimiento y selección de servicios. El capítulo uno presenta la introducción.

El capítulo dos presenta una breve reseña sobre cloud computing y composición de servicios. Su fin es establecer el marco teórico de los recursos cloud y sus implicaciones en un proceso de composición de servicios. Se estudia además, la transición del esquema convencional SOA a cloud computing y su convergencia en la composición.

El capítulo tres presenta el problema de selección, el concepto de condiciones y criterios de calidad de servicios(QoS) y la revisión de literatura de los diferentes enfoques de mecanismos de selección. Su fin es establecer la conceptualización formal de QoS y estudiar las tecnologías,áreas de estudio y estrategias computacionales de los enfoques de selección, con el propósito de definir los puntos de convergencia teórica de Ar_WSDS y sus factores diferenciadores y de innovación que serán expuestos en los capítulos siguientes.

El capítulo cuatro, constituye el modelamiento y creación de Ar_WSDS. Después de considerar brevemente la noción de recursos cloud, el problema de selección y los criterios QoS de un servicio atómico y compuesto; se detalla la formalización lógica matemática de la propuesta de solución a partir de un sistema de reconocimiento de patrones. En él se definen las estrategias

de reconocimiento y el proceso de ranking o valoración de servicios.

El capítulo cinco, corresponde a la etapa de experimentación y obtención de resultados. Tras haber desarrollado Ar_WSDS se establecen los escenarios de experimentación para analizar su eficacia en el proceso de descubrimiento y búsqueda de servicios. Adicional, se compara el enfoque propuesto con dos estrategias de selección y ranking de servicios.

Finalmente, el capítulo seis presenta las conclusiones, en él se discute el desarrollo de la investigación y los trabajos futuros en el área de selección de servicios y reconocimiento de patrones.

CAPÍTULO 2

Generalidades de cloud computing y de la composición de servicios

La composición del servicio Web consiste en la combinación de servicios existentes para satisfacer los requerimientos u objetivos de los clientes, para esto es necesario asignar recursos adecuados a un conjunto de Servicios que constituyen un servicio compuesto o complejo[9].

El paradigma de Cloud [1] computing surge como una arquitectura orientada a servicios (SOA) que trabaja con los protocolos y estándares que ella ofrece, su ventaja principal es la dar a los usuarios y proveedores una cantidad de recursos disponibles de forma distribuida que exponen sus funcionalidades a través de servicios[10]. En el entorno cloud, los requerimientos de los clientes en la etapa de análisis y diseño de la solución son de vital importancia, ya que es necesario dar a los consumidores y proveedores los recursos más acordes a los objetivos que se deben alcanzar[11]. En este punto identificamos dos actores, el cliente o usuario que expresa un objetivo alcanzar y el proveedor que interactúa con la plataforma cloud y da los pautas para la implementación de servicios de software.

En este capítulo se describe los fundamentos de los servicios web(WS) , composición y su interacción con ambientes SOA y Cloud. Así mismo se realiza una arquitectura genérica de la composición en cloud y se enuncian las bases de la selección y búsqueda de WS, objetivo primordial de este tesis doctoral.

2.1. GENERALIDADES

El concepto de cloud computing fue dado inicialmente por el NIST(National Institute of Standards and Technology), donde se definió como:

Ün modelo que permite el acceso bajo demanda y a través de la red a un conjunto de recursos compartidos y configurables (como redes, servidores, capacidad de almacenamiento, aplicables y

[1] El lector podrá identificar esté concepto por su término en español *nube*.

servicios) que pueden ser rápidamente asignados y liberados con una mínima gestión por parte del proveedor del servicio"[12].

Al ser un modelo, su objetivo es servir de referencia para el diseño y construcción de infraestructuras que ofrezcan recursos que puedan ser fácilmente gestionados. Sus características principales se ven enmarcadas en las operaciones de: Autoservicio bajo demanda, Múltiples formas de acceder a la red, Compartición de recursos, Elasticidad y medición de servicio.

Sus características giran en el concepto de "recursos configurables", así en el autoservicio bajo demanda un usuario puede acceder a todos los recursos que él necesite. Las formas de acceder al recurso remotamente se pueden realizar a través de cualquier medio y dispositivo de comunicación. Se comparten recursos conforme el cliente solicite o personalice. Los servicios pueden crecer dependiendo de las condiciones del negocio del usuario y finalmente, es posible medir y controlar el consumo de un recurso computacional.

Las ventajas de la utilización de este modelo han sido ampliamente estudiadas[13] y en [14] donde se explican las estrategias para la implementación de Cloud Computing. Estos estudios destacan la importancia de la adopción de estas tecnologías y cómo las empresas de TI se ven enfrentadas a un cambio paradigmático, ya que el modelo ofrece una forma de gestión de recursos a un mínimo esfuerzo por parte de los proveedores de Cloud Computing y de acuerdo a las necesidades de los clientes. Su paradigma conjuga diferentes maneras de integración de infraestructuras hardware, ambientes heterogéneos de desarrollo y grupo de aplicaciones que trabajan bajo el concepto de demanda y de pago por uso[15].

En [12] se definen dos modelos computacionales de trabajo para la cloud, estas comprenden:

- Modelo de despliegue: Se refiere a las infraestructuras lógicas y físicas que los proveedores ofrecen a sus clientes, tomando en cuenta el tipo de red(LAN-WAN) y el consumo promedio de recursos que irán a consumir. El despliegue puede ser visto como:
  - Clouds privadas: La infraestructura y aplicaciones residen en la compañía. Su particularidad principal es que a nivel de desarrollo e implantación de software son regulados por las licencias software y no al pago por uso o demanda. La ventaja de este modelo de despliegue es que el centro de información y datos se pueden integrar nuevas herramientas sin la intervención de un proveedor[16].
  - Clouds públicas: La infraestructura y aplicaciones residen en los proveedores. Su acceso es gestionado de manera elástica y poseen altos niveles de escalabilidad. Los proveedores pueden ofrecer a sus clientes diferentes planes de acuerdo a los servicios que requieren las compañías. Los sistema de administración de aplicaciones son dados por los proveedores, esto conlleva a que su monitoreo y actualización sean más eficientes[17].
  - Clouds híbridas: Comprende la unión de dos o más clouds, por ejemplo privada y pública, cuya convivencia se da manera autónoma y comparten recursos a través de diferentes tecnologías[18]
- Modelo de Servicio: Comprende los servicios que son ofrecidos en cloud, estos son:
  - Software as a Service(SaaS): Proporciona al cliente la capacidad de usar aplicacio-

nes que se ejecutan en una infraestructura cloud, por lo general, a través de una aplicación web. Para el cliente es transparente aspectos como servidores, red, sistemas operativos, almacenamiento. El cliente paga por el consumo de recursos q la aplicación realice[19].

- Plataform as a Service (PaaS): Proporciona al cliente la capacidad de desplegar y construir sus aplicaciones en una infraestructura en la cloud. El cliente administra esta infraestructura a través de APIs especializadas. No supervisa la arquitectura física cloud, solo realiza acuerdos de utilización de plataforma. Tiene la ventaja de poder utilizar software licenciado, y de tener mantenimiento y actualización de estas herramientas de manera transparente por parte del proveedor[16].
- Infrastructure as a Service (IaaS): Proporciona al cliente la capacidad de administrar infraestructura física y lógica como procesamiento, almacenamiento, conexión entre redes Y otros recursos de una infraestructura computacional con pago de su función de uso. A través de conceptos como "virtualización", el proveedor permite que el cliente despliegue y ejecute Software, que puede ser sistemas operativos, servicios y Aplicaciones de propósito general o especifico. El cliente controla los sistemas operativos, el almacenamiento, y la Figura 1 muestra la interacción de los modelos de despliegue y servicio, de esta forma, un tipo de cloud puede ofrecer cualquiera o los tres tipos de servicios y un cliente puede escoger qué tipo de despliegue es el que necesitan las aplicaciones desplegadas[19].

La Fig.2.1, muestra la interacción de los modelos de despliegue y servicio, de esta forma, un tipo de cloud puede ofrecer cualquiera o los tres tipos de servicios y un cliente puede escoger que tipo de despliegue es el que necesita.

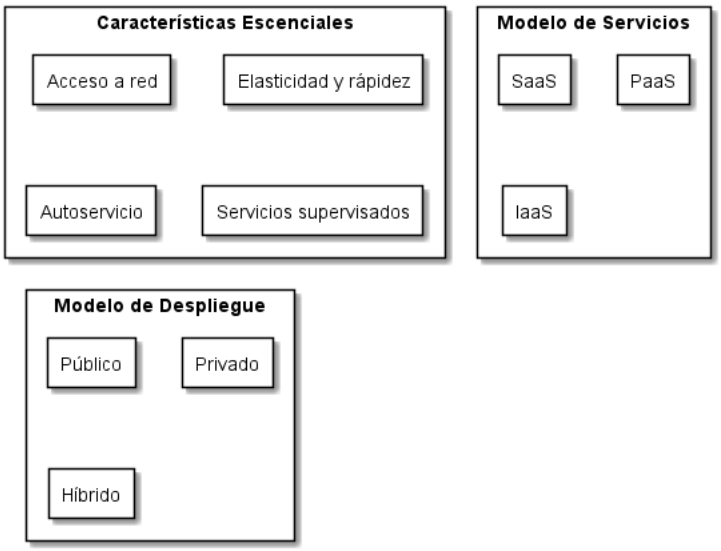


Figura 2.1: Vista general de operación de tecnologías Cloud Computing. Adaptado de: [20].

Dentro del marco proporcionado por Cloud se encuentra un modelo de trabajo orientada a servicios que si bien no corresponde a especificaciones técnicas iguales, si comparten caracte-

rísticas en común que son relevantes a la hora de trabajar con el concepto de servicio. En la sección 2.2, se irá a tratar lo concerniente a SOA y Cloud desde el punto de vista arquitectónico y cómo sus modelos son complementarios.

2.2. CONVIVENCIA ENTRE SOA Y CLOUD

Como paradigma de trabajo la Cloud ofrece un entorno atractivo a los clientes ya que ellos pueden operar con infraestructuras y recursos computacionales de fácil gestión, dando la ventaja de escalar dinámicamente a través de la virtualización de esos recursos en comparación a los paradigmas actuales[16].

No obstante, SOA (Services Oriented Architecture) aparece como una arquitectura para organizar y reutilizar componentes en forma de servicios. Ella comprende su implementación, la definición de procesos empresariales y los mecanismos para visualizar resultados y que usan comúnmente servicios web que son consumidos por los clientes y administrados por los proveedores, gran parte de esas aplicaciones empresariales consumen recursos SOA[21]. De esta forma SOA, es en gran parte uno de las arquitecturas más usadas por la versatilidad en la reutilización de servicios.

SOA, comprende una arquitectura de software para el desarrollo de aplicaciones TI, cuyo principal objetivo es la reutilización de servicios con funcionalidades predefinidas(es decir, se conoce a priori su función y características), independiente y desacoplada, cuya interacción es dada a través de APIs conjugadas en interfaces que son expuestas para que otros clientes y/o proveedores que puedan acceder a ellas para combinar servicios y crear otros de mayor complejidad. Estos servicios complejos pueden ser consumidos con herramientas que ayudan a definir la lógica de un proceso de negocio o simplemente ser instanciados por otras aplicaciones, actividad que concierne al objeto de estudio de esta tesis “composición de servicios” como un logro relevante ofrecido por SOA.

A nivel de implementación, las aplicaciones en el modelo SOA ofrecen un alto nivel de acoplamiento en comparación a los desarrollos convencionales, que si bien, ofrecen enfoques basados en patrones de diseño, el trabajo con SOA trae consigo la orquestación del software como servicio permitiendo una fácil integración, al ser SOA una arquitectura abierta basada en composición la orientación de servicios ofrece componentes y/o unidades interoperables e independientes de las plataformas de despliegue, fácilmente identificados en colecciones llamadas directorios e implementados como servicios Web[16].

En términos generales, se puede decir que SOA comprende un paradigma (o conjunto de prácticas) arquitectónico para procesos de negocios que pueden operar bajo plataformas heterogéneas con diferentes proveedores[15], exponiendo su funcionalidad a través de servicios que ofrecen bajo acoplamiento y que son fácilmente reutilizables por cualquier proceso de negocio interno o externo.

Las aplicaciones en SOA son construidas generalmente bajo servicios compuestos, o con la

Tabla 2.1: Arquitecturas tecnológicas en SOA

Arquitectura	**Definición**
Composición de Servicios	La arquitectura de un conjunto de servicios reunidos en composición de servicios.
Inventario de Servicios	Arquitectura que soporta un conjunto de servicios relacionados que son independientemente estandarizados y gobernados
Empresarial Orientada a Servicios	La arquitectura de la propia empresa (para cualquier tamaño de empresa) orientada a servicios.
Arquitectura de Servicios	Se trata de la arquitectura de un único servicio.

Fuente: Adaptado de [22]

integración de cualquier recurso o plataforma sobre diferentes arquitecturas. Resulta primordial ofrecer mecanismos de reutilización que aseguren que estos servicios puedan ser consumidos por el negocio, y comprender el modelo de arquitecturas tecnológicas que SOA ofrece, para tener claro el contexto de operación del servicio que se desea implementar y/o usar. La tabla 2.1 presenta las arquitecturas convencionales en un ambiente SOA.

Cloud computing comprende un modelo de gestión de recursos computacionales bajo demanda, con interacciones transparentes del cliente y cuyo proveedor se encarga de medir y cobrar por el consumo que realice de ese recurso, sus ventajas fueron analizadas en el punto anterior (véase sección 2.1).

Bajo el modelo cloud es indispensable que los proveedores cuenten con herramientas de gestión donde el cliente pueda fácilmente escalar recursos y medir su consumo actual. Esta característica hace indispensable que indistinto del tipo de servicio que ofrezca la cloud, cuente con plataformas operacionales con altos niveles de configuración de sus componentes, así como una correcta comunicación con el hardware que operan.

La tabla 2.2, presenta una caracterización de similitudes encontradas durante la revisión de literatura basado en los trabajos de [11, 19, 23-25] entre SOA y Cloud Computing:

En efecto, SOA y Cloud tienen puntos de convergencia relevantes como se puede analizar en la tabla 2.2, lo que significaría que modelos de trabajo en SOA pueden convivir y utilizarse con Cloud. Es decir, dentro de una arquitectura de TI el diseño y desarrollo de aplicaciones orientada a servicios pueden operar sobre PaaS, su infraestructura física centrada en modo de IaaS, de modo tal que traería agilidad en la implementación e implantación de nuevas aplicaciones(perspectiva de SOA) con una ventaja de escalabilidad, rendimiento y gestión dado por la Cloud.

Tabla 2.2: Caracterización entre SOA y Cloud Computing

Característica	**SOA**	**Cloud Computing**
Servicio	Unidades o componentes de código claramente definidas donde se exporta su interfaz de uso	Estrategias para ofertar recursos (SaaS, PaaS, IaaS)
Procesos de Negocio	Se modelan a través de unidades interalacionadas que facilitan la acción de ser compartidas dentro y fuera del negocio. Permite orquestar procesos y colocarlos en workflow para su posterior integración.	Se modelan como pequeños servicios de negocio, con capacidades de escalabilidad, heterogeneidad de plataforma y acceso flexible. Su actividad se ve principalmente en la forma de SaaS.
Aplicaciones de terceros	Comprenden un conjunto de aplicaciones colocadas en forma de catálogo, donde las empresas pueden adicionar o componer nuevas aplicaciones o procesos de negocio para un dominio local o externo.	Son ofrecidas en modo SaaS, de forma tal que se cuenta con un conjunto de proveedores que distribuyen software y se encargan de su administración.
Plataforma	Ofrecidas por componentes definidos por BPM o de infrasestructura como ESB(Enterprise Service Bus) y EAI(Enterprise Application Integration).	Son ofrecidas en modo PaaS, donde técnicamente la plataforma operacional es administrada por un proveedor de cloud.
Desarrollo nativo de aplicaciones	Es realizado en las empresas con herramientas licenciadas o a través de proveedores. La infraestructura puede ser local o remota.	Son ofrecidas en el modo de IaaS. Las empresas usan recursos virtualizados y comparten herramientas de desarrollo del proveedor.

2.3. ARQUITECTURA ORIENTADA A SERVICIOS PARA CLOUD COMPUTING

SOA y cloud son modelos independientes, pero cuya convivencia como se ha venido mencionando tienen implicaciones favorables para las empresas de IT y para sus clientes, por ende, cloud ofrece un valor agregado a SOA. El desarrollo de aplicaciones utilizando cloud, no irá a reemplazar el uso de tecnologías de sistemas distribuidos e integración proporcionado SOA, factor relevante que conlleva a que las empresas dedicadas al IT sigan construyendo sistemas con altas capacidades de integración sobre SOA, pero que necesiten satisfacer requerimientos no funcionales para que las aplicaciones sean más eficientes, por lo tanto, la demanda de recursos computacionales tendrá un comportamiento creciente.

Las trabajos realizados por [26] y [27] son un referente de arquitecturas que integran a SOA y Cloud Computing, en ellos se establece una arquitectura inicial denominada *Cloud*

Computing Open Architecture o por sus siglas en inglés (CCOA) donde expone siete principios arquitecturales basados en diez componentes para integrar SOA con las tecnologías software y hardware de virtualización proporcionada por los ambientes cloud computing. Su trabajo establece cómo se pueden separar y combinar los procesos de negocio encapsulados en servicios hacia un ambiente cloud que ofrece la infraestructura, el software y las aplicaciones necesarias para que estos servicios operen adecuadamente.

CCOA ofrece una arquitectura reutilizable y escalable a través de módulos que representan diferentes proveedores de cloud y que el cliente en cualquier momento puede cambiar, concepto ligado a SOA. Los principios de CCOA comprenden:

1. Administración del Ecosistema: Se refiere a la interfaz entre clientes y proveedores de cloud computing. Consta de todos los procesos de administración de los recursos y como van ser ofertados y medidos a los clientes.
2. Virtualización: Concierne todo lo referente a la virtualización de hardware y software en ambientes cloud computing.
3. Reutilización de Servicios: Concierne a los procedimientos y formas de desarrollar servicios que puedan ser reutilizados. Estos servicios son colocados bajo tecnologías web services.
4. Suscripción cloud: Se refiere a los procesos y perfiles de suscripción de clientes en los proveedores clouds, algunos pueden ser o no pagos. La suscripción define la capacidad que un cliente tiene de utilizar recursos computacionales y su tiempo de acceso.
5. Configuración de modos de trabajo en cloud: El cliente puede escoger que tipo de servicio usar: SaaS, IaaS o PaaS o un conjunto de todas ellas.
6. Representación de la información: La información de los servicios en cloud deben contar con un estándar de representación, así cualquier cliente puede entender los servicios ofrecidos por un proveedor cloud, esto con el fin de modelar de datos, por lo general en XML como estándar de facto. Esto obliga a que cada proveedor deba estructurar su información de tal forma que la arquitectura la entienda y por ende cualquier cliente que interactúe con ella.
7. Gobierno y Calidad de servicio cloud: Asegura que todos los servicios y recursos ofrecidos por los proveedores de cloud estén funcionado adecuadamente. Se deben implementar herramientas para monitorear y modificar recursos cuando uno no cumpla con una restricción de calidad reportada por un cliente.

Por otra parte el trabajo realizado por [27],ofrece una especificación teórica en cuatro capas que integran tecnología cloud con una arquitectura convencional SOA denominada SOCCA, ellas comprenden:

- Capa de Proveedores cloud: Se definen cuáles son los proveedores y los recursos computacionales que tiene disponible para sus clientes.
- Capa de Ontología Cloud: Establece una especificación de información a través de un lenguaje ontológico de los servicios y proveedores cloud. Ofrece un nivel de estandarización que permite tener información para ser tratada por las directrices que un cliente use de SOA a través de la capa de Cloud Broker.

- Capa de Cloud Broker: Ofrece información detallada de los proveedores, de precios, de parámetros de calidad, hardware, software y la descripción de los servicios prestados.También ofrece un sistema de puntaje que permite a los clientes analizar qué tan fiables es un proveedor y un histórico de precios según recursos ofrecidos.
- Capa de SOA: Define la forma en que se van usar los cloud operados desde la visión de SOA, es decir, Los proveedores no alojan los servicios, en lugar de ello publican los servicios en unidades claramente definidas e interpretadas por la capa de ontología para su posterior replica y distribución en los diferentes proveedores de la cloud. Así mismo, los desarrolladores especifican los requisitos no funcionales que el servicio debe cumplir pudiendo decidir que proveedor cloud cumple con esos criterios.

El trabajo de Tsai et al.[27] tiene la ventaja de establecer la comunicación entre capas y de separar las funciones de cloud y de SOA. De igual manera, explica como el diseño de las aplicaciones es basado en el modelo de reutilización de servicios basados principalmente en tecnologías de servicios web y las ventajas de tener composición servicios para requisitos altamente complejos. La perspectiva de la cloud solo la da a nivel de cuál es la mejor arquitectura de despliegue y que recursos satisfacen la correcta ejecución de un servicio.

2.4. SERVICIOS WEB

2.4.1. Definición

Los servicios Web(por sus siglas en inglés WS) comprenden un conjunto de aplicaciones bajo una arquitectura cliente-servidor que utilizan para el intercambio de información el Protocolo de Transferencia de Hipertexto (HTTP). Su objetivo de construcción reside en el hecho de servir como solución a problemas de interoperación, así los servicios web permiten que aplicaciones que se ejecutan en plataformas heterogéneas puedan intercambiar y procesar datos. Su información de descripción así como de intercambio de mensajes es realizado a través de formatos XML[28], facilitando la interoperabilidad y extensibilidad. El concepto de Servicio web no es igual al de servicio en la perspectiva de SOA, por le contrario, el servicio web se refiere a la unidad de código en forma de componente con características de reutilización e independencia, denominados *servicios web atómicos* cuya combinación con otros WS lleva a la creación de *servicios web compuestos o moleculares* que realizan tareas complejas(servicios que atienden más de un requerimiento funcional).

Conceptualmente, un servicio es un componente de software que ofrece una interfaz para que aplicaciones u otros servicios pueden utilizarlo. El consumidor y el proveedor de servicios se comunican a través del intercambio de mensajes de la forma petición-respuesta, haciendo transparente para el consumidor los recursos computacionales que el proveedor del servicio usa para procesar su solicitud. Específicamente, el WS es una entidad de carácter “abstracta” cuya funcionalidad es expuesta a través de la publicación de sus servicios y esta es invocada por un por un agente. El agente constituye la parte de lógica funcional computacional de un programa base que usa el WS para enviar y recibir mensajes, según sea el caso. Su carácter de

heterogeneidad hace que un agente pueda ser escrito en diferentes lenguajes de programación que consumen el mismo WS.

Adicional, el agente identifica cual es el proveedor del WS que puede ser una persona(Cliente) o una organización, en el primer caso es llamado solicitante o consumidor y en el segundo entidad proveedora. Esta diferenciación conceptual ofrece la separación lógica entre el programa base que invoca el servicio y su implementación interna, basados en el concepto de encapsulamiento ofrecido por lenguajes orientados a objetos [25].

Los servicios web son descritos inicialmente descritos en WSDL (Web Services Description Language). Este permitía que los proveedores describieron los servicios y los publicaran en repositorios utilizando información en un estándar denominado *Description Discovery and Integration* o por sus siglas en inglés (UDDI), cabe resaltar que existen más formas de describir servicios en un repositorio, sin embargo la más utilizada es UDDI. De esta forma el cliente o consumidor del servicio, realizaba una búsqueda por el repositorio, este era seleccionado y se proporcionaba su información utilizando WSDL. El consumidor invoca el servicio web con datos WSDL y el proveedor de servicio da la respuesta de su servicio web.

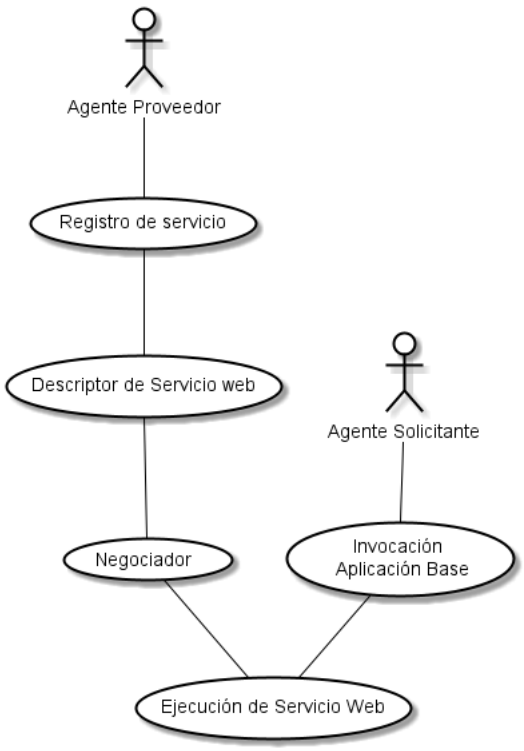


Figura 2.2: Operación de un servicio web

.

La descripción del servicio define el formato y la estructura de los mensajes, de igual forma especifica los protocolos de transporte y la forma en que se va a llevar a cabo la serialización. Esto con el fin de establecer una comunicación libre de errores y de inconsistencias semánticas entre el cliente y el proveedor.

La Figura 2.2 presenta el funcionamiento general de un WS y de sus entidades, se establece como punto de partida que la entidad solicitante conoce a priori la localización del proveedor que debe "registrar" los WS a través de la UDDI y describe el WS a nivel de la funcionalidad que esta expuesta para ser consumida ("descriptor"). La acción del "negociador" consiste en establecer los acuerdos de uso que tiene el WS , así como sus requerimientos no funcionales (parámetros QoS). Adicional, se proporcionan los mecanismo de serialización para que el solicitante y el proveedor puedan interactuar. Los intercambios de mensajes se realizan con un estándar usando SOAP o REST (ver tabla 2.3 en el apartado siguiente) y permiten que la entidad proveedora "ejecute" y de respuesta al WS invocado por el solicitante.

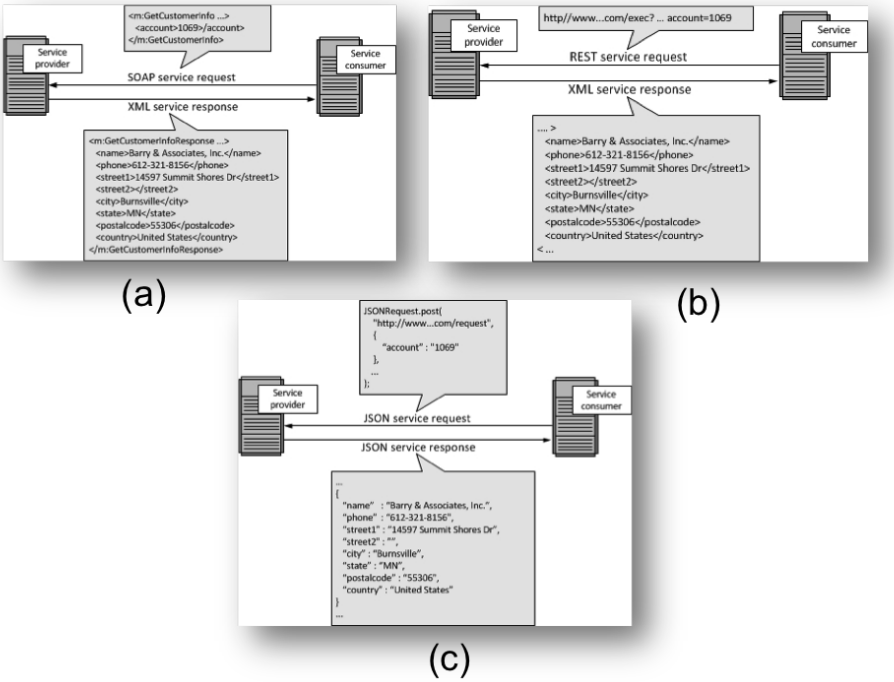


Figura 2.3: Ejemplo de SOAP, REST y JSON. **Fuente:** [29].

2.4.2. SOAP y REST

SOAP y REST son dos especificaciones para el intercambio de mensajes ampliamente usados por los WS. La Figura 2.3 muestra la interacción de cada una de las especificaciones, en el ítem (a) se muestra un ejemplo típico de SOAP. Su información está descrita por su cabecera en la petición que realiza al proveedor (descripción del nodo: getCustomerInfo) y su respuesta dada en el body (name, phone, etc…) y como se puede observar en formato XML con sus respectivos nodos padres e hijos. El ítem (b) un ejemplo de REST, cliente realiza una petición con la URI del servicio y la información de la petición es devuelta en un XML, símil a la de SOAP. En el ítem (c) el formato JSON presenta su información en conjuntos "pares-valor", cabe resaltar que este tiene la desventaja en comparación de SOAP de documentar la estructura de su interfaz de JSON,

a diferencia de SOAP que utiliza WSDL y estructuras XML [30]. JSON ofrece las ventajas de su fácil integración a las aplicaciones web sin la adición de un lenguaje intermedio que realice el correspondiente llamado del servicio.

En consecuencia, SOAP usa solamente XML para ejecutar su intercambio de mensajes, el uso de formatos XML han resultado más útiles que los desarrollos antiguos como (DCOM) y el mismo (CORBA), que dependían exclusivamente de transacciones binarias teniendo una dependencia de interoperabilidad que fallaba en entornos altamente heterogéneos. No obstante, las estructuras XML usadas por SOAP pueden llegar a ser complejas, los lenguajes computacionales actuales permiten realizar invocaciones y lecturas de datos a través de sus API. Pese a estas implementaciones, la operación es manual aumentado el riesgo del error en el formateo de la estructura de información que se envía como la que se recibe, ya que XML es altamente restrictivo a nivel sintáctico [31]. A pesar de esto la especificación WSDL es fácilmente asociada con SOAP y dan una clara descripción del funcionamiento del WS, esta ventaja ha sido ampliamente usada por las IDEs actuales que traen herramientas de diseño automatizadas para crear fácilmente referencias al WS y construir dinámicamente las clases (en el caso de orientación a objetos) que irán a operar con el WS.

El manejo de excepciones en SOAP es automático, si ocurre un problema se puede verificar la solicitud y manejar exactamente donde ocurrió la excepción, adición de gran valor para el WS que como componente de software no cuenta con esa propiedad de análisis de errores [32]. SOAP cuenta con una tabla de códigos de errores con su respectiva descripción que facilitando el trabajo de los desarrolladores en la corrección de problemas.

Por otra parte, REST es una alternativa más ligera, no usa XML. Es un enfoque para obtener exclusivamente información bajo una URL usando las acciones definidas por el http: GET, POST, PUT y DELETE. En contraste con SOAP, REST puede invocar WS en diferentes formatos de información como CSV, JSON y RSS, siendo estos formatos de fácil manejo por cualquier lenguaje computacional de alto nivel [34].

En conclusión, la mejor manera de eligir entre SOAP y REST depende la funcionalidad que se le de al WS, ya que aunque REST es de fácil manejo por las invocaciones al HTTP y la fácil acoplación de sus datos en las aplicaciones, SOAP posee un manejo de errores y de extensibilidad de operaciones que no tiene REST.

2.4.3. Descripción de Arquitectura

Una arquitectura de WS proporciona una especificación para la interoperabilidad entre diferentes aplicaciones de software que se ejecutan en plataformas. Se define un marco común de trabajo para tener un modelo conceptual y un contexto estándar para trabajar los WS y sus las relaciones entre sus distintos componentes [25]. La ventaja principal de la arquitectura es describir la forma como pueden interactuar los WS, sin imponer restricciones de diseño, al contrario ofrece unas características de nivel superior para que cualquier WS indistinto de su lenguaje computacional base pueda interactuar con otros y con las entidades solicitantes[35].

Tabla 2.3: Formatos de especificación de servicios web

Especificación de Servicios Web	**Descripción**
SOAP	Protocolo Simple de Acceso a Objetos. Se basa en XML y permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja a través de los protocolos de la capa de aplicación del modelo TCP/IP. Su estructura es sencilla, se compone de dos capas header (cabecera) que contiene la información de transmisión del mensaje a nivel de red. El body (cuerpo) que contiene la especificación del mensaje que irá a ser consumido por el servicio web.
REST	Representational State Transfer. Comprende un conjunto de principios arquitectónicos en el que los servicios web describen los recursos que van usar, detallando su forma de accederlos y los mecanismos de transferencia usados en HTTP. Tiene la gran ventaja de usar HTTP con su definición formal para crear, leer, actualizar y borrar recursos. La definición de recursos la hace por su URI convirtiéndose en una estándar de facto.

Fuente: Adaptado de : [33]

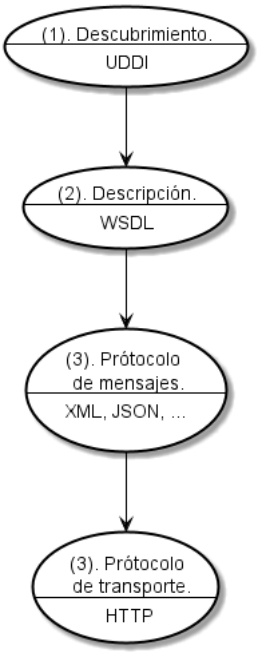


Figura 2.4: Arquitectura de un Servicio Web. **Fuente:** Adaptado de [36].

La Figura 2.4 presenta la arquitecturas por capas de un WS, a continuación se describen:

Capa 1. Descubrimiento: Capa responsable de centralizar la publicación de la información funcional del WS en un registro común de una forma ágil y sencilla. Actualmente, es realizado a través de *Universal Description, Discovery e Integration* (UDDI). Propuestas de investigación recientes se han interesado en adicionar capas con el fin de minimizar el coste computacional de las búsquedas de WS.

Capa 2. Descripción: Nivel donde se describe la interfaz pública de un WS. Es manejada a través del *Web Service Description Language* (WSDL).

Capa 3. Protocolos de mensajes: Su función es la de proveer un estándar para el intercambio de mensajes entre los solicitantes y los proveedores (ver tabla 2.3).

Capa 4. Protocolos de Transporte: Capa responsable de transportar los mensajes entre las aplicaciones, principalmente son usados los protocolos de la capa de aplicación del modelo tcp/ip como http, smtp y ftp. Actualmente algunos sistemas manejan comunicaciones por *Blocks Extensible Exchange Protocol* (BEEP), este es comúnmente usado en aplicaciones p2p, su ventaja radica en que a partir de el se pueden crear protocolos para aplicaciones en red[37].

2.4.4. Servicios Web y Cloud

Las tecnologías propias de servicios web son derivadas de SOA, sin embargo, como se analizó en la sección 2.2, la interacción de SOA con Cloud son un complemento atractivo para las nuevas arquitecturas de IT. Es indispensable reconocer que la visión de trabajo es de desarrollos

bajo el modelo SOA que utilizan los recursos de Cloud computing. Allí nace una diferencia conceptual a nivel de "servicio" , para Cloud un servicio es definido por SaaS, IaaS o PaaS o denominado SIP (conjunción de sus siglas) y para SOA un "servicio web" es un tipo de servicio con características de reutilización, empaquetamiento, independencia, heterogeniedad de invocación que intercambia mensajes a través del protocolo HTTP. Así bien, un SIP puede utilizar tecnologías de "servicios web" para operar con sus recursos y un desarrollo en SOA podría eventualmente sus "servicios web" interactuar con recursos Cloud Computing. La Figura 2.5 presenta un esquema basado en diagramas de Venn que ilustra el gobierno de los servicios web en entornos SOA y Cloud.

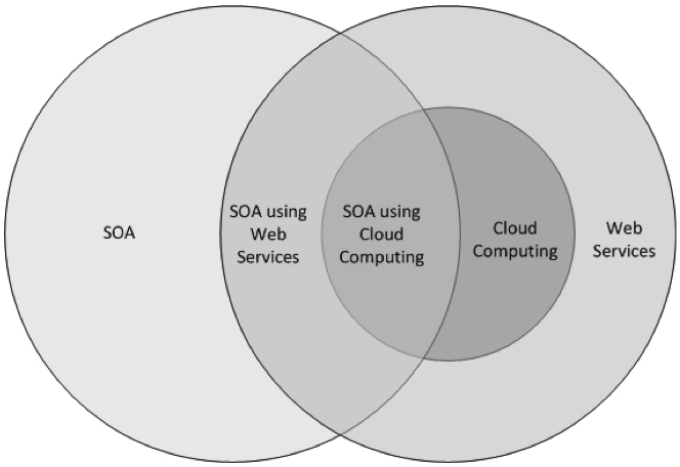


Figura 2.5: Relación de tecnologías de servicios web con SOA y Cloud.
Fuente: [29]

Se puede notar en la Figura 2.5 la forma en que las tecnologías de servicios web engloban las operaciones de arquitecturas Cloud y SOA (ver sección 2.3), no obstante, se pueden tener desarrollos en SOA que usen otro tipo de servicios.

2.5. COMPOSICIÓN DE SERVICIOS

2.5.1. Perspectiva SOA

SOA trae consigo el concepto de WS, a nivel de diseño propone servicios que sean visibles y que puedan integrarse con otros de la manera más sencilla, propiedades que son dadas en sistemas que son capaces de exportar sus interfaces de uso en una semántica que sea fácilmente consumible por otros servicios (Protocolo de mensajes)[38], ventaja que ha llevado a la creación de aplicaciones distribuidas con soporte multilenguaje. No obstante, los requerimientos actuales de los sistemas de información hacen necesario el desarrollo de sistemas de negocio complejos que operan con un número considerable de WS, que son englobados en una operación denominada *Composición de Servicios Web(WSC).*

Si bien los WS son una forma de proporcionar comunicación entre componentes y/o aplicaciones de software heterogéneo y su utilización se ve prometedora; su interacción a nivel atómico (un único servicio) sólo converge en aplicaciones de un dominio específico; es decir, a satisfacer requerimientos básicos de una(s) aplicación(es), no obstante esta solución no siempre resulta útil en las necesidades de los sistemas de información actuales, donde se requiere de un alto dinamismo y comunicación entre las diferentes partes involucradas en una actividad económica para obtener información de toma de decisiones [39]. Esto conlleva a tener formas de interacción entre diferentes servicios atómicos en proveedores distribuidos que ofrezcan estrategias de intercambio de mensajes y colaboraciones, solucionar un requerimiento de alto nivel, denominándose este tipo de unidades de código: Servicios complejos [39, 40].

Sin embargo, pese a sus ventajas la implementación de WS atómicos son desarrollados con tecnologías diversas, por lo que las tecnologías de integración y composición requieren gran esfuerzo para estandarizar operaciones. Estos inconvenientes se han convertido en retos de investigación en el área de composición de servicios Web, donde se busca crear estándares para el desarrollo de herramientas y lenguajes que faciliten la integración de WS.

La Figura 2.6, presenta los componentes involucrados en un sistema de composición de servicios web basados en las especificaciones genéricas proporcionadas por la W3C [41]. El sistema inicia su actividad cuando un cliente solicita un requerimiento (paso 1) y el desarrollador analiza cuales servicios son los que que hacen parte del workflow y lo modela en una lenguaje de especificación (paso 2) como WSBPEL(“Web Services Business Process Execution Language”) [42], BPML (“Business Process Markup Language”), WSFL (“Web Services Flow Language”), entre otros.

Estos lenguajes tienen la facultad de abstraer las operaciones de los servicios y colocarlos en flujos de trabajo donde se genera una lógica de negocio que soporta la solución del requerimiento entrante. Los servicios involucrados deben ser encontrados en los diferentes repositorios (paso 3), para esto cada uno de ellos deben exponer su información a través de un lenguaje de descripción de servicios como WSDL(“Web Services Description Language”), WSCI (“Web Service Choreography Interface”), entre otros. En el proceso puede ocurrir que dos o más servicios ofrezcan funcionalidades símiles por lo que se debe escoger el mejor servicio tarea prevista en el paso 3. Los WS son analizados según el requerimiento no funcional que deben cumplir (paso 4). Finalmente, con los servicios escogidos y depurada su funcionalidad, se colocan en el motor de ejecución(paso 5) y se crea el servicio compuesto que será invocado por alguna aplicación base, el motor cuenta con mecanismo que aseguran el funcionamiento adecuado del sistema en el momento de la ejecución (tolerancia a fallos, monitoreo, control de excepciones, etc).

La composición puede ser estática o dinámica; en la primera, los procesos de integración se llevan a cabo con actividades proporcionadas por el desarrollador de software, esto es, él debe realizar el despliegue y la coordinación manualmente (colocar servicios en una secuencia lógica de interacción-flujos de trabajo), enfoque que conlleva a un tiempo de trabajo considerable. En la segunda, se establece un nivel de dinamismo, tanto en la operación del sistema, como en los procesos involucrados en una composición (búsqueda, análisis de requerimientos no

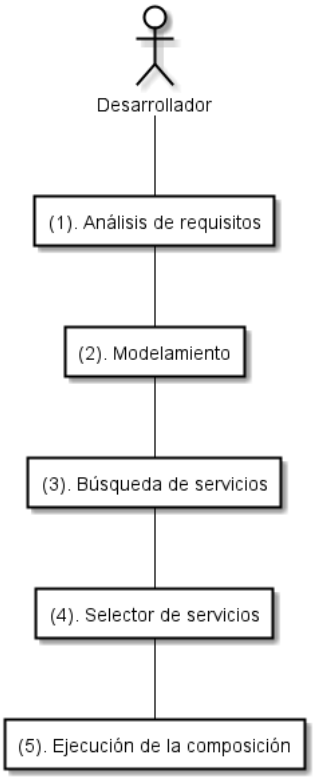


Figura 2.6: Vista general de un Sistema de Composición de Servicios Web.

funcionales, entre otras) [43].

En una composición estática, el desarrollador interviene de manera directa eligiendo los servicios que quiere utilizar y la forma de programar las interfaces para la comunicación entre ellos, su resultado es un archivo bajo una especificación de un lenguaje donde se describe su secuencia de ejecución. En contraste, en una composición dinámica donde el sistema automáticamente busca, selecciona y ejecuta los servicios que satisfagan el requerimiento. Estos son analizados para extraer información de servicios involucrados y se procede a través de ellos la identificación de relaciones, flujos y servicios involucrados, con este resultado se modela en una semántica que ofrezca parámetros ser especificados por lenguaje formal, con esta descripción los motores de búsqueda y selección, escogen los servicios adecuados según disponibilidad(Fase de Descubrimiento) y son pasados al motor de ejecución para la obtención de resultados. Esencialmente, en estos sistemas la representación del servicio y su definición en los repositorios son de vital importancia.

El proceso de composición de servicios se agrupa en dos actividades básicamente: el primero corresponde al diseño y Modelado y el segundo a la selección, ejecución y monitoreo del proceso de composición [44], de esta manera a partir de los componentes del sistema presentados en la Figura 2.6, se pueden agrupar dentro de dos procesos básicos como se presenta en la Figura 2.7.

Los Servicios web son componentes independientes de software publicados por los pro-

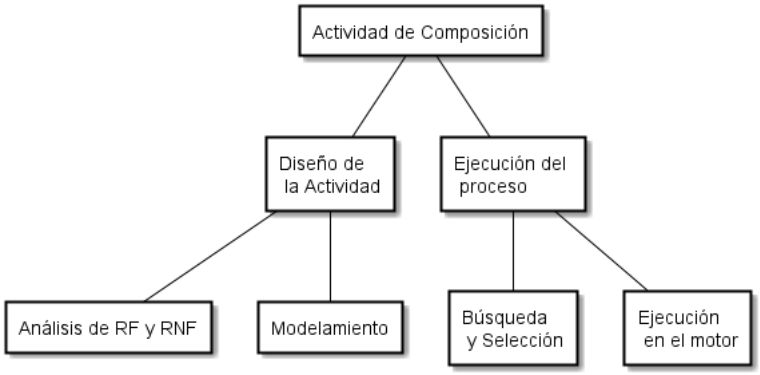


Figura 2.7: Actividades generales del proceso de composición.

.

veedores de servicios e invocadas por los solicitantes para una determinada funcionalidad, en muchos casos, esta funcionalidad puede ser otorgada por más de un servicio dentro de uno o más repositorios, surgiendo problemas algorítmicos y de arquitectura, como búsquedas, análisis de disponibilidad del servicio, entre otros parámetros necesarias para que una composición se lleve a cabo de manera eficaz[45].

En SOA todos estos procesos de composición, así como la gestión y administración de sus servicios se realizan con los Enterprise Service Bus(ESB), cuyo objetivo es proveer una plataforma de virtualización para que los clientes no operen directamente con un servicio, proporcionado un nivel de encapsulamiento para el cambio ágil de interfaces y desacoplamiento entre los servicios expuestos y los servicios que serán consumidos por los clientes[44].

Los ESB se implementan utilizando tecnologías relacionadas con los middleware, cuya función es la de proveer comunicación e interacción transparente con aplicaciones, hardware y sistemas operativos todos ellos heterogéneos, en términos generales, ofrece una API para que las aplicaciones puedan utilizar funciones de un sistema de manera sencilla [46]. La composición de servicios tiene dos maneras de combinar servicios: la orquestación y la coreografía términos usados para referirse a la forma como interactúan los diferentes servicios web; sin embargo, su operación es complementaria, es decir, en una proceso de composición pueden existir las dos [47]. Su tarea se ve reflejada en el contexto de servicios que son expuestos por la organización y son consumidos internamente y de aquellos servicios utilizados para procesos de negocios de alto nivel que requieren interacciones de muchos servicios en varias organizaciones. En la orquestación el proceso de negocio es manejado por un controlador central que también puede ser un servicio Web y tiene como funciones: coordinar interacciones asíncronas, controlar los flujos de trabajo, gestionar las transacciones de negocios y monitorear los procesos de negocio [48]. La coreografía, no depende de un coordinador central, el servicio participante sabe exactamente cuándo debe activarse y con quién interoperar. Se basa en el principio de la colaboración y es utilizado principalmente para intercambiar mensajes en los procesos de negocios públicos.

La composición dinámica como ya se ha mencionado, pretende automatizar en gran medida

las operaciones que realiza el desarrollador desde el momento de la especificación, diseño del modelo y la escogencia de los servicios, para esto se requiere que el desarrollo de los WS compuestos ofrezcan una especificación clara que incluya desde su perspectiva atómica [35]:

1. La descripción de los resultados: define las capacidades del servicio, los resultados que son dados a los usuarios, sus parámetros iniciales de funcionamiento y el detalle de las limitaciones de aplicabilidad. Define principalmente sus precondiciones.
2. Funcionalidad: Esto es, una descripción sobre el cómo se debe invocar el servicio y sus características posteriores a la ejecución.
3. Acceso: Corresponde, a la descripción de los protocolos de comunicación y los detalles del formato del mensaje utilizado para el intercambio de información.

En una composición estática, la búsqueda y escogencia de servicios se lleva a cabo en la etapa de diseño, se analizan las interfaces de los componentes (comunicación entre servicios y paso de mensajes) para su posterior despliegue. Los proveedores involucrados son conocidos y difícilmente son cambiados ya que los requerimientos son claramente definidos[44]. No ofrece flexibilidad en cuanto a tener requisitos cambiantes, o en la posibilidad de alterar algún servicio por otro que constituya una mejor alternativa. A nivel de tolerancia a fallos es débil debido a que en la gran mayoría de escenarios problemas se debe reconstruir el diseño de la composición, retrasando el soporte en ejecuciones de tiempo real.

Por otra parte, la composición dinámica busca, selecciona y reemplaza un servicio de forma automática en tiempo de ejecución, útiles en dominios donde los requerimientos son altamente cambiantes; sin embargo, su trabajo resulta complejo ya que aspectos como control de errores, tiempo de respuestas, entre otros, cobran gran importancia siendo en la actualidad un área de investigación en ejercicio [43]. De forma ideal, la automatización inicia desde el desarrollador de aplicaciones de servicios Web, donde el sistema le facilita dinámicamente las interacciones entre un proveedor de servicios y un consumidor, ofreciendo en lo posible flujos alternativos de composición con posibilidades de ser cambiantes a partir de requisitos no funcionales que deben ser adaptados por los WS, adicional ofrecerá la lectura y formas de invocación de interfaces de descripción de servicios en tiempo real cuando un proveedor realice algún cambio, siendo independiente de plataformas y lenguajes, sin preocuparse por su implementación interna.

2.5.2. Perspectiva Cloud Computing

El propósito fundamental de entregar una serie de servicios independientes es que se puedan combinar en composiciones de servicios totalmente funcionales capaces de automatizar la tareas de negocio más grandes y complejas. Cada composición de servicios tiene una arquitectura de composición de servicios correspondiente[22], así los WS son una tecnología de facto para la creación de aplicaciones que requieren altos niveles de reutilización, por su naturaleza y principios de construcción, son el pilar del desarrollo de software en SOA, paralelo a esto y con la convivencia de SOA con cloud las aplicaciones distribuidas en estos entornos y con el paradigma de WS están siendo utilizadas ampliamente. La ventaja de construir aplicaciones bajo servicios compuestos y expresados en lenguajes formales para composición hacen de esta forma

de trabajo una tarea atractiva para los desarrolladores, que ven un sin numero de posibilidades para la ejecución de servicios compuestos en ambientes cloud.

Debido a que estos tipos de especificaciones de arquitecturas de servicios están amenudo protegidas (según las necesidades planteadas por el principio de abstracción de Servicios), una arquitectura de composición sólo podrá hacer referencia a la interfaz técnica y acuerdo de nivel de servicio (o sus siglas en inglés SLA), publicado como parte del contrato público de servicios. Un *Service Level Agreement es un contrato entre un proveedor de servicios de TI y un cliente, en el que especifica los servicios que el proveedor debe ofrecer*. El contrato constituye uno de los ítems más importantes especialmente para que el cliente pueda monitorear el desempeño del proveedor a partir de datos medibles[49]. En la cloud los SLA no están fácilmente visibles por parte de los proveedores, en contraste, las entidades solicitantes se ven abocadas a trabajar de manera automática para la escogencia y medición de los recursos que necesitan, un reto importante de investigación es la creación de SLA en una semántica que sea fácilmente consumibles por motores de búsqueda y que garanticen que la utilización de un WS no viole los requisitos de los SLA[50]. Los SLA son una forma de minimizar los riesgos entre cliente y el proveedor, ya que es aquí donde se especifican gran parte de los requerimientos no funcionales que un servicio debe cumplir y satisfacer, su principal objetivo es el de establecer tanto para el consumidor como el proveedor estrategias de utilización de los servicios.

Desde el punto de vista de SOC,la gestión de la composición tiene una característica no funcional que es el proveedor, de esta forma los servicios son ligeramente acoplados, ya que un servicio podría acoplarse en aplicaciones complejas (aplicaciones que se desarrollan a través de una composición de servicios) y por ende, si existiese un fallo podría ocasionar une efecto en cadena ya que pueden tener aplicaciones externas que hagan uso de ese servicio complejo. En adición, un servicio puede estar definido en un flujo de trabajo y este en tener copias en varios proveedores, por lo que a nivel de procesamiento de computo , la operación de búsqueda es todo un reto, a diferencia de SOA donde muchos de los repositorios son centralizados, en cloud estos se encuentran de manera distribuidas y descentralizados [51]; por consiguiente, el monitoreo y gestión de los sistemas construidos bajo esta premisa son temas de gran importancia.

Un conjunto de servicios en cloud, además, podrían ser utilizados en un entorno de aplicación SOA, esta gran integración resulta en un modelo de trabajo ambicioso que combina las ventajas de los dos enfoques con la singularidad principal de ofrecer servicios en plataformas distribuidas con recursos elásticos y todo estos accedidos a través de protocolos de internet. Sin embargo, desde la perspectiva de gobierno de SOA surgen problemas importantes a la hora de considerar su trabajo, afirmación ampliamente explicada por [15]:

"La coexistencia de ambos entornos no es tan sencilla, sobre todo en lo que se refiere a todos los factores que componen el gobierno de SOA, una cuestión extremadamente importante ya que permite a las organizaciones mantener continuamente actualizados la planificación, diseño, validación, publicación, provisión, monitorización, modificación, optimización y securización de los entornos distribuidos. De este modo, se garantiza que los servicios desplegados en los entornos de aplicación de empresa –estén construidos en cloud, en mainframes o en cualquier otra plataforma– satisfacen los requerimientos corporativos en cuanto a métodos operacionales, políticas y cumplimiento normativo.

Pero el modelo cloud computing introduce perturbaciones en el gobierno SOA que podrían ser de gran trascendencia".

De forma que aunque cloud puede ofrecer toda la infraestuctura física y lógica para el correcto funcionamiento de servicios web en aplicaciones desarrolladas bajo el modelo SOA, el mantenimiento de las plataformas y proveedores requieren un manejo integral que resulta complejo cuando son los proveedores los que ofrecen herramientas de gestión y la cantidad de soluciones en el mercado es variada, imposibilitando una medición de efectividad de servicios en cloud pueda ser objetiva.

2.5.3. Arquitectura Genérica

La composición de WS en cloud consiste en múltiples proveedores de servicios y un WS puede integrarse a partir de múltiples proveedores sin importar características como ubicación, plataformas y velocidades de ejecución[52], se elige los WS interconectar por diferentes proveedores de servicios sobre el objetivo que se debe alcanzar por un proceso empresarial en particular.

Un servicio web compuesto es creado a partir de múltiples servicios web atómicos y compuestos, que interactúan de acuerdo flujo de ejecución o secuencia lógica que define el objetivo alcanzar por la composición, si bien existe múltiples formas de representar la composición de WS, es propósito para esta tesis que su modelo se basa en un grafo acíclico (ver Figura 2.8), donde la combinación de cada uno de los WS involucrados en los nodos convergen en alcanzar el requerimiento dado por la entidad solicitante, tanto funcional como no funcional. El grafo se puede representar a través de relaciones algebraicas que incluyen secuencias, flujos,WSitch y ciclos.

Cada vértice del grafo representa un servicio en la composición. Cada arista de salida de un nodo (servicio) representa las salidas y las postcondiciones producidas por el servicio. Cada arista entrante de un nodo representa las entradas y precondiciones del servicio. Un servicio en cualquier etapa de la composición puede tener potencialmente como sus entradas todas las salidas de sus predecesores, así como las entradas de la consulta. Los servicios en la primera etapa de la composición solo pueden utilizar las entradas de la consulta. La unión de las salidas generadas por los servicios en la última etapa de la composición debe contener todas las salidas que la consulta requiere que se produzcan. También las postcondiciones de los servicios en cualquier etapa de la composición deben implicar las precondiciones de los servicios en la siguiente etapa.

Cada servicio web atómico y/o compuesto constituye una unidad de software que atiende parte del proceso de negocio. De esta forma, dado un requerimiento R existirán n WS atómicos o compuestos en m clouds que deben seleccionarse para alcanzar un objetivo O dentro de unos parámetros de calidad (QoS), estos parámetros son una colección de requerimientos no funcionales para cada WS y por la composición que deben ser procesados en el menor en un tiempo posible, basándose de la premisa de fiabilidad del tiempo de respuesta del usuario.

Un workflow es representado en la Figura 2.8 a través de un grafo , las líneas representan los

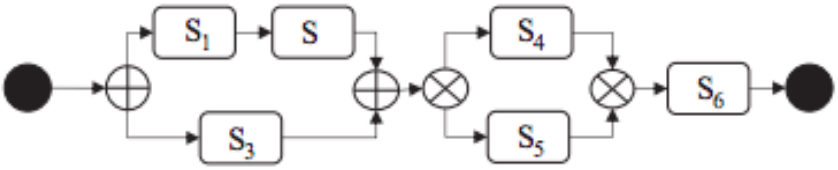


Figura 2.8: Workflow de un proceso de Composición.
Fuente: [52]

llamados o invocaciones de cada servicio (aristas de entrada o salida), los ⊕ simboliza que un flujo puede darse en un camino o bien por otros flujos y el símbolo ⊗ representa la finalización de un flujo para actividades individuales, finalmente, el símbolo • representa inicio y fin de la composición. De está forma:

1. Estado inicial definido por el requerimiento ”R”.
2. Los vértices representan un WS necesario para alcanzar ”O”. Cada WS tiene un vector de QoS dados en los SLA de cada cloud.
3. Las aristas de salida de un nodo representan las postcondiciones.
4. Las aristas de entrada las precondiciones.
5. Estado final definido por la meta ”O”.

De manera ideal, cada nodo antecesor sus postcondiciones deben satisfacer las precondiciones de los nodos sucesores, en términos de desarrollo de software, cada WS debe tener los resultados de las interfaces de los WS que le suceden en la composición, esto comprende: tipos y cantidad de datos, y firmas de operaciones que correspondan a nivel semántico con lo solicitado en R.

Comunicar WS entre diferentes clouds es una operación costosa, por lo que encontrar el mínimo número de clouds con los WS adecuados es un problema de investigación abierto[1, 24, 53, 54], sin embargo, en la actualidad existen propuestas de optimización en la fase de selección y descubrimiento que serán objetos de estudio para esta tesis.

La Figura 2.9 presenta la arquitectura genérica del proceso de composición en ambientes cloud, para esto se tomo como punto de partida las figuras 2.6 y 2.7 que definen un sistema de composición en ambientes SOA, no obstante, no se realiza la distinción de los comportamientos estáticos o dinámicos ya que como se ha venido analizando el comportamiento dinámico comprende ejecución automática de los procesos de “modelado” y “búsqueda y selección” del WS. El proceso de composición inicia cuando una entidad usuario posee un requerimiento funcional que debe ser desarrollado por un sistema donde intervengan WS atómicos (paso 1), seguido existe una actividad de interpretación de ese requerimiento para generar el modelo de composición (paso 2) a partir de los WS existentes en los diferentes entidades proveedores cloud, cabe resaltar que un proveedor puede tener WS idénticos en diferentes clouds con distintos parámetros QoS (problema de selección y búsqueda objetivo primordial de esta tesis). Cuando los WS ya están seleccionados se establece un formato de especificación servicios donde estará registrada los datos de los WS , con esta información un motor especifico de composición ejecuta el modelo y entrega resultados a la entidad solicitante(usuario).

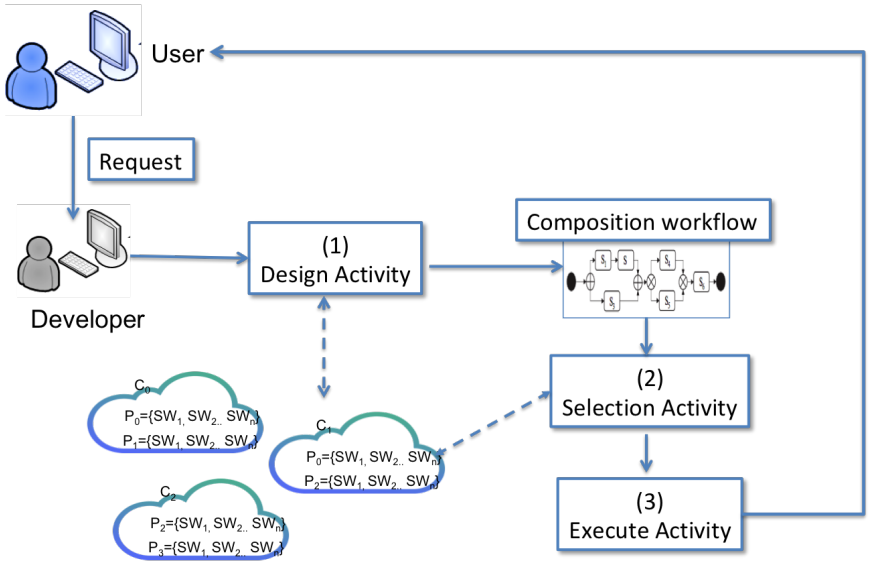


Figura 2.9: Arquitectura Génerica de la Composición en Cloud. Fuente: Elaboración propia

Es necesario establecer convenciones de entradas y salidas de cada etapa de la arquitectura para tener un marco de referencia para modelar computacionalmente el sistema (desarrollo del sistema a nivel de software), a continuación son descritas:

1. De Usuario a Desarrollador: La entrada al sistema se realiza con el requerimiento del usuario, esta solicitu encapsula las peticiones funcionales y no funcionales. La parte no funcional comprende los parámetros QoS por WS, por proveedor y por el servicio final compuesto.
2. Desarrollador a Diseño: El desarrollador o en su defecto en ambientes dinámicos el analizador de requisitos será el encargado de modelar conceptualmente los WS involucrados en el workflow, para esto se hace necesario que conozca la información de los WS de los diferentes proveedores y sus respectivos SLA.
3. Selección a cloud: Si bien se pueden tener los WS claramente establecidos cumpliendo las normas QoS, los workflows pueden ser variados. La fase de diseño puede generar más de un modelo con niveles de optimización similes. La actividad de selección tiene el objetivo de escoger los WS que esten operando con un tiempo de respuesta aceptable desde la entrada al sistema hasta la ejecución, es decir, pueden seleccionarse WS en un momento de tiempo especifico pero en el momento de su ejecución sus requisitos no funcionales hallan cambiado.

2.6. CONCLUSIONES

La selección y búsqueda de un servicio es una tarea compleja debido al hecho de la disponibilidad de recursos tanto en etapa de diseño o de ejecución, y su comportamiento puede ser incierto y dinámico. Actualmente, las investigaciones concernientes a la composición de servicios se

enfrentan a desafíos que subyacen de la idea básica del como buscar y seleccionar un servicio en un tiempo eficiente[3].Adicional a esto la naturaleza dinámica del cloud han llevado a que la especificación de requerimientos no funcionales sea dinámica y por ende un modelo de composición puede cambiar desde la fase del diseño hasta su ejecución.

La composición dinámica es una tarea muy compleja por este motivo solo algunas fases de la actividad de composición son automatizadas, su forma de modelar es variada como reporta la literatura, sin embargo, predomina el modelo a través de grafos que permite tener una abstracción clara de los WS involucrados.

CAPÍTULO 3

Selección y búsqueda de servicios web

La composición de servicios depende de los WS atómicos involucrados en el modelo de composición, siendo necesario algoritmos que aseguren la correcta acoplación y funcionamiento de los WS, cada uno de los cuales cuenta con diferentes condiciones de calidad(QoS), por tanto la selección de los servicios que cumplan con los requerimientos no funcionales exigidos por las entidades clientes es una tarea de gran atención en aplicaciones basadas en composición.

En este capítulo se describe el problema de selección y los mecanismos reportados por la literatura de heurísticas de solución, de igual forma se realiza un acercamiento del modelo de trabajo de selección de WS.

3.1. DESCRIPCIÓN DEL PROBLEMA DE SELECCIÓN

La integración eficaz de WS en entornos de aplicaciones comerciales y procesos de negocio son un reto para los desarrolladores de software que se ven enfrentados a seleccionar WS adecuados que no violen el modelo y sus requisitos. Debido a la naturaleza dinámica y crecimiento exponencial de los proveedores de WS, las aplicaciones compiten por seleccionar los mejores WS que optimicen recursos computacionales de modo que sin una gestión adecuada de servicios las aplicaciones podrían experimentar errores funcionales y atenuaciones críticas en rendimiento de los recursos ofrecidos por las diferentes plataformas cloud.

Si bien el crecimiento de clientes ha sido exponencial en este modelo, el problema radica en la cantidad de WS que pueden ser compuestos para satisfacer un requerimiento en particular [5], es difícil para los usuarios analizar las opciones que se adapte a sus necesidades. Más aun en el momento de implementar un software bajo este modelo, tareas como la búsqueda, mediación y seguimiento de un WS cobran gran importancia[55].

En un sistema de composición de servicios en cloud, se seleccionan y acoplan WS para generar un flujo de trabajo, cada tarea del flujo puede tener un conjunto de WS candidatos con funcionalidad similar, y estos servicios web tienen diferentes parámetros de calidad de servicio(QoS).

Es indispensable entonces, crear mecanismos para seleccionar dinámicamente los mejores servicios de acuerdo con la QoS para permitir que el modelo de composición alcance su objetivo. Finalmente, esta situación conlleva a un problema de optimización de cómo seleccionar los servicios web para cada tarea de modo que sean respetados las condiciones QoS de la composición.

El desarrollo de aplicaciones con WS compuestos inicia cuando se diseñan los procesos de negocio orientados a resolver los requisitos de las entidades solicitantes, el desarrollador en la medida de lo posible intenta reutilizar los servicios existentes para construir los diferentes procesos. La eficiencia de este enfoque de desarrollo SOA se debe en gran medida a la disponibilidad de WS públicos y gratuitos en aumento[56]. Infinidad de proveedores de internet tienen en sus repositorios un numero considerable de WS accesibles con SOAP o REST principalmente.

No obstante, Sidhu et al.[57] refiere que pese a lo prometedor que pueda parecer, muchos proveedores no son claros al momento de describir la información QoS del servicio aumentado el riesgo que el proceso de selección falle, lo que conlleva al aumento de carga laboral del desarrollador debido a que tiene que verificar en detalle cada unos de los servicios para asegurar su correcto funcionamiento y acoplamiento.

Por esta razón la selección de un WS con niveles QoS dentro de los rangos establecidos por los requerimientos no funcionales iniciales son una tarea de gran responsabilidad, un error a este nivel ocasionaría no solo fallos en la funcionalidad esperada, sino también un incremento en los costos económicos y de uso de recursos. Por este hecho, las propuestas de investigación tanto comerciales como académicas se han visto evocadas a proponer mecanismos de selección y búsqueda óptimas que ayuden a minimizar el riesgo y aumentar la calidad de los procesos de negocio construidos en esta forma de desarrollo.

La búsqueda y selección de un WS en entornos estáticos consiste en la consulta manual de estructuras de información que usan como persistencia de datos los registros UDDI. Los resultados dependen de la veracidad y fiabilidad de la información almacenada, hecho que es ideal en operaciones donde el conjunto de servicios es limitado y no existen cambios funcionales radicales. Sin embargo, la realidad de la aplicaciones empresariales es otra, y su entorno es altamente dinámico, donde los WS varían y por consiguiente sus atributos de calidad resultando ineficiente este enfoque.

El área de conocimiento de calidad de servicio es comúnmente usada para el análisis y toma de decisiones de las características no funcionales de los WS y es el núcleo de trabajo para la selección de servicios[58]. Las condiciones QoS comprenden un conjunto de propiedades tomadas de los requerimientos no funcionales del WS como el de su proveedor, de esta forma, existirán propiedades que deben ser medidas en el cliente y otras en el proveedor. El análisis de estas condiciones es un factor decisivo para medir el comportamiento de un proveedor, adicional a esto, se puede obtener información de ranking del servicio basados en estadísticas como su facilidad de uso, rendimiento, cantidad de entidades solicitantes, entre otras características.

Finalmente, con lo anterior expuesto, el problema de selección de WS para una actividad de composición es definido como:

Dados los requerimientos funcionales y no funcionales(QoS) de un modelo abstracto de composición y una especificación de proveedores cloud y WS cada uno con sus respectivas condiciones QoS, se debe seleccionar el(los) WS necesarios para que el modelo de composición alcance su objetivo.

3.2. CONDICIONES QOS

Conceptualmente la calidad de servicio de un WS se analiza desde el contexto de la calidad de software, donde el término se refiere, al cumplimiento de los requisitos funcionales, a la identificación de defectos y al análisis de su diseño. En [59] se define la calidad como: *Un conjunto de técnicas cuyo objetivo es hacer que coincidan las necesidades de solicitantes y proveedores de los servicios, ambas basadas en los recursos de red disponibles.* El concepto denota las dos entidades ya estudiadas con anterioridad, los solicitantes y los proveedores. De esta forma es necesario establecer la diferencia que la ingeniería del software ofrece para el manejo de requerimientos, dividiéndolos en aspectos funcionales y no funcionales. En el primer aspecto se evalúa que el WS cumpla con la funcionalidad para el cual fue creado, y desde el punto no funcional se refiere a las condiciones QoS consignadas y operadas desde los acuerdos de nivel de servicio. De esta manera formalmente se puede definir una condición QoS para un WS como aquella característica necesaria para ofrecer una funcionalidad de acuerdo a unos términos descritos en un SLA. En ambientes cloud, un WS puede tener diferentes valores de QoS en un grupo de proveedores, es decir, el mismo WS puede estar replicado en distintas cloud con diferentes SLA.

Es indispensable establecer la importancia de la selección de WS para una composición desde la visión de estos dos tipos de requerimientos. En estos términos, una condición funcional a nivel de los WS candidatos para el modelo de composición usando workflows comprende la dependencia de las firmas de las operaciones sean las mismas al nodo sucesor y cuyos datos de salida correspondan a los parámetros de entrada del servicio receptor; es decir, la definición sintáctica de las interfaces del WS deben ser iguales, de manera general esta definición se encontraría en el archivo WSDL respectivo.

Tomando como referencia los estudios en [3, 10, 60, 61], los aspectos de calidad de servicio mayormente considerados son: disponibilidad, tiempo de respuesta, rendimiento, fiabilidad e integridad, presentados en la Tabla 3.1 donde se definen cada uno de estos criterios.

3.3. FUNCIONAMIENTO DE LA SELECCIÓN

El mecanismo de selección es implementado a partir de la comprensión de las requisitos funcionales y condiciones QoS involucradas en el modelo de composición, de esta forma y como se definió en el problema de composición, las condiciones exigidas están relacionadas desde la visión del solicitante y otra del proveedor, idealmente la estrategia de selección será capaz de involucrar las dos entidades y hacer una búsqueda para encontrar cuales son los WS que cum-

Tabla 3.1: Condiciones generales de calidad de un servicio Web

Condición	**Descripción**
Disponibilidad	Un WS debe estar listo para su uso inmediato o en un momento determinado. La disponibilidad también está asociada con la disponibilidad del tiempo de reparación (TTR) cuando un servicio ha fallado y que indudablemente se espera que sea durante un tiempo corto.
Accesibilidad	Se puede expresar como una medida de probabilidad, que indica el porcentaje de éxito o de posibilidad de una creación de instancias de servicios de éxito en un punto en el tiempo. Es difícil saber las situaciones en que un WS está disponible, pero no es accesible.
Integridad	El WS debe mantener la exactitud de los datos en la interacción con respecto a la fuente, y la correcta ejecución de las transacciones
Rendimiento	Se mide en términos de desempeño y latencia. Un mayor rendimiento y los valores de latencia más bajos representan un buen desempeño. El rendimiento se puede representar como el número de solicitudes a WS, asistidas en un periodo de tiempo determinado. La latencia es el tiempo que tomó prestar el servicio, desde el envío de una solicitud hasta la llegada de la respuesta.
Fiabilidad	Comprende el número de fallos por mes o año puede ayudar a llevar un control y una medida de la fiabilidad de un servicio Web; también puede referirse a la seguridad en cuanto a la entrega de mensajes enviados y recibidos por los solicitantes de servicios y por los proveedores de servicios.
Regulación	Los WS se basan en una variedad de estándares como SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration) y WSDL (Web Services Description Language). Es necesario que los proveedores de servicio cumplan estrictamente las versiones correctas de los estándares para que los solicitantes invoquen adecuadamente los WS
Seguridad	Es la confidencialidad y la autenticación correcta de las partes involucradas, los mensajes de cifrado y el control de acceso proporcionado por los prestadores del WS. El proveedor de servicios puede tener distintos enfoques y niveles de prestación de seguridad en función del solicitante del servicio.

Fuente: Adaptado de : [33]

plen con estos criterios. Tomando en cuenta esto, el funcionamiento es descrito de la siguiente forma:

1. Registro de los requerimientos de la entidad solicitante: En este punto se debe especificar formalmente una interfaz y una estructura de información con los requerimientos funcionales y condiciones QoS.
2. Registro del modelo de composición: Especificación del workflow y los WS de la composición.
3. Registro de proveedores: Especificación de los clouds disponibles con sus WS y respectivos archivos SLA.
4. Componente de Negociación: Es un mecanismo de software que analiza los requerimientos funcionales y las condiciones de QoS y a través de estrategias algorítmicas determina si un WS es candidato para pertenecer al modelo de composición.
5. Componente de Selección: Es un mecanismo software que a partir de los resultados obtenidos por el componente de negociación, selecciona de los WS candidatos aquellos que satisfacen la composición, en esté punto el servicio creado es denominado molecular o servicio compuesto (WS_C).

La siguiente lista establece la especificación formal de las entidades en un proceso de selección:

1. Proveedores de WS: Un proveedor define sus servicios en archivo de especificación de servicios (WSF), de forma: $WSF = \{WS_1, WS_2, WS_3, .., WS_n\}$ donde cada WS_i corresponde al WS desplegado en ese proveedor.
2. Cloud (C): Una cloud puede contener un número finito de proveedores que exponen sus WSF, de forma: $C = \{WSF_1, WSF_2, WSF_3, .., WSF_n\}$, donde "n" es la cantidad de proveedores de WS.
3. Servicio Web (WS): Un WS se define como un conjunto de datos de entrada (I), de salida(O) y sus requerimientos no funcionales (RNF). La especificación de estos datos se dan conforme a los estándares de WS y que son fácilmente identificables en los directorios de registro de cada proveedor. Por lo tanto, un WS es una 3-tupla <I,O,|RNF|>y |RNF| puede ser un vector de RNF.
4. Service Level Agreement (SLA): Define los acuerdos para usar los servicios entre el cliente y el proveedor, su función es especificar y clarificar las expectativas del funcionamiento, establecer la responsabilidad, y detallar las alternativas y consecuencias si el funcionamiento o la calidad del servicio no son los acordados por ambas partes. Por lo tanto, para cada WSC se debe cumplir un SLA y de forma que : $SLA = \{RNF_1, RNF_2, RNF_3, .., RNF_n\}$, donde RNF son los requerimientos no funcionales de cada servicio . Estos RNF están contenidos en la 3-tupla de la definición del WS.

La Figura 3.1, presenta el esquema conceptual de proveedores y servicios, en él se puede observar como un proveedor puede tener múltiples WS y cada uno de ellos tiene un conjunto de RNF que definen sus criterios de calidad; de esta forma, el problema de selección deberá abarcar estrategias que ofrezcan la clasificación más eficaz, ya que podrán existir replicas de un WS en proveedores con valores distintos en sus QoS.

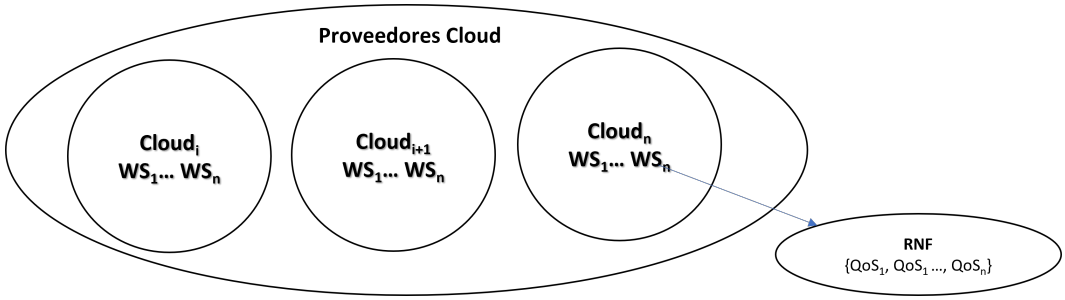


Figura 3.1: Proveedores Cloud y Servicios Web

.

3.4. MECANISMOS DE SELECCIÓN

Existen diferentes mecanismos de selección descritos en la literatura principalmente en el área de SOA[38] y [25] en ambientes cloud, siendo de nuestro interés este último basados en el problema descrito en la sección 3.1. Así la composición de servicios web generalmente sólo pueden encontrar las secuencias de composición en una sola nube desaprovechando el evento que ese WS en particular pueda encontrarse con condiciones satisfactibles en un ambiente multi-nube.

Para realizar la revisión sistemática de los mecanismos de selección reportados por las diferentes investigaciones, se utilizan las ideas reportadas en Vera [62] y se adapta para el contexto de la investigación. Su proceso consiste en técnicas bibliométricas y de selección basado en la teoría de la moda, que es definida como:

"La creencia temporal de un grupo de personas acerca de lo benéfica que puede resultar una innovación, bien sea su implementación en la práctica o su uso como objeto de investigación"[62, 63].

En relación a lo anterior, el análisis documental comprende la especificación de criterios de búsqueda, una ventana de tiempo y la tendencia de investigación o moda, esté trabajo denomina a esté proceso de análisis CVM.

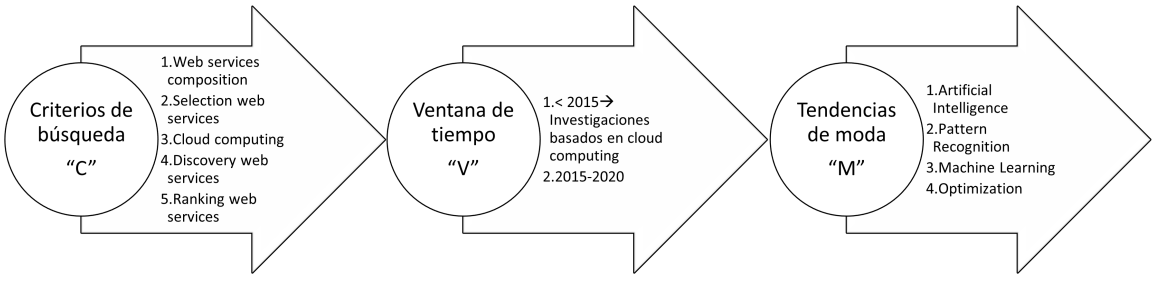


Figura 3.2: Modelo para revisión sistemática de literatura

.

La Figura 3.2 y la Ecuación. 3.1 definen el proceso para la selección de enfoques. Cada parte del análisis está referenciada por una letra C, V o M ; acompañado de un índice que presenta su valor canónico presentado en la Tabla 3.2.

Tabla 3.2: Convenciones para el análisis documental CVM

Convención	**Valor**
C1	Web services composition
C2	Selection web services
C3	Cloud computing
C4	Discovery web services
C4	Ranking web services
V1	ventanas de tiempo entre 2011-2015 basados en SOA
V2	ventanas entre 2015-2021 basados en cloud computing
M1	Artificial intelligence
M2	Patter recognition
M3	Maching learning
M4	Optimization

$$Selected\ approach = ((C1 \vee C2 \vee C4 \vee C5) \wedge C3)) \wedge (V1 \oplus V2) \wedge (M1 \vee M2 \vee M3 \vee M4) \quad (3.1)$$

Con la aplicación del modelo CVM, se establecen los siguientes enfoques que aportaron a la solución formal de Ar_WSDS, a continuación son descritos tomando el orden presentado en las tendencias de moda.

Una propuesta de búsquedas de servicios y composición en cloud es presentada en [64] basada en el algoritmo de búsqueda Cuckoo a través de técnicas para satisfacción de restricciones. Sus escenarios de prueba se basan en entornos de nubes distribuidas con réplicas de WS con diferentes valores QoS. Este trabajo aporta una visión de la utilización de multinubes con sus datasets respectivos, destacando la utilización de filtros para el proceso de análisis sintáctico y otro para la parte funcional de cada WS candidato a ser seleccionado por el modelo de composición que debe satisfacer. Así mismo en [65] es presentado un método de selección basado en ontologías, este trabajo formaliza el WS para permitir la descripción exacta de sus interfaces y parámetros, agilizando de manera circunstancial el proceso de selección a nivel sintáctico.

En [66] se propone un método para descubrir y seleccionar servicios usando aprendizaje automático. El sistema predice los nuevos valores QoS a partir de la información que se encuentra en su lenguaje de descripción de servicios web(WSDL). Su análisis se basa en la obtención de datos directamente de los WSDL ya que en él se describe la forma de cómo componer una solicitud de servicio web y describe la interfaz que proporciona el proveedor para consumirlo. Esté análisis permite filtrar aquellos WSs que están disponibles en un momento determinado.

Un trabajo que desarrolla la noción de descubrimiento de patrones y clasificación de WS es propuesto en [67], a través de técnicas de minería de datos para clasificar los servicios a través de ciertos requerimientos no funcionales, su estudio da como resultado que los principales puntos de análisis en un WS lo constituyen el tiempo de respuesta, el rendimiento y su disponibilidad , métricas claves en una actividad de composición.

El trabajo propuesto en [68] utiliza un método de descubrimiento de servicio basado en redes de Petri, en él se desarrolla todo un sistema de composición modelado sobre un cluster de servicios en tiempo real, su sistema dota al WS de un nivel semántico que permite etiquetarlo con información del análisis de sus valores QoS, con el objetivo de tener agrupamientos de servicios según un requerimiento no funcional en particular.

En [69] los autores presentan un algoritmo para la selección de WS en repositorios con réplicas de servicios. Su estrategia computacional se basa en la teoría de relaciones de equivalencia y modela un sistema para almacenar servicios que evite la redundancia en su semántica de registro. Sus experimentos se basan en modelos de composición secuenciales, esto es donde cada servicio encaja o se conecta con su servicio predecesor.

El trabajo realizado en [6] enfatiza el problema de composición en entornos cloud en el momento de seleccionar WS con la descripción de escenarios con replicas de ellos. Su trabajo desarrolló un algoritmo y modelo matemático usando heurísticas de conocimiento-aprendizaje basado en evolución diferencial y se basa en búsquedas en tres criterios QoS: tiempo de respuesta, disponibilidad del servicio y fiabilidad. Sus escenarios de prueba utilizan datos aleatorios para verificar el funcionamiento del algoritmo. Este punto es importante ya que los dataset en este campo son escasos y muchos trabajos optan por crear sus propios conjuntos de datos.

En [53] se desarrolla una técnica de indexación para la gestión de la información de proveedores cloud y sus servicios. El proyecto implementa algoritmos para la búsqueda y selección de servicios en un ambiente controlado, su estrategia de solución es basada en la clasificación de proveedores por parámetros QoS y categorización de servicios por su funcionalidad. En particular, su desarrollo propone la utilización de una estructura de información dinámica que tiene la ventaja de modelar los datos de los proveedores, así como sus requisitos no funcionales y codificarlos de tal forma que el algoritmo de selección encuentre fácilmente similitudes entre servicios y proveedores, ofreciendo la ventaja de realizar la clasificación de una manera óptima. SMICloud [70] es un framework para la búsqueda y selección automática de proveedores de cloud en función de requisitos de usuarios preestablecidos. El framework implementa negociaciones entre SLA de forma dinámica, además desarrolla una técnica para la clasificación de servicios usando métricas para cada parámetro QoS, dando como resultado una categorización de servicios por cada proveedor, a través de un mecanismos de evaluación estándar de los servicios en la nube para cada tipo de QoS.

Gomes, *et al.* [71] presenta un middleware para la gestión de recursos cloud basado en RNF de servicios preestablecidos por los usuarios. La investigación ofrece una perspectiva sobre el problema de ejecución de aplicaciones a nivel de la adaptación según la demanda de nuevos requisitos que pueden surgir durante una actividad de composición, específicamente su

estudio analiza el cómo los recursos informáticos pueden ser accesados a través de una IaaS con su infraestructura de cómputo virtualizado. La propuesta se basa en técnicas model-driven que permiten especificar los requerimientos no funcionales de un servicio y seleccionarlos en múltiples cloud. Su resultado es una alternativa de selección bajo la premisa de costos reducidos de consumo de recursos.

En [72] se realiza un modelo para WSC sobre SaaS que toma en cuenta requerimientos QoS y los requisitos no funcionales presentados por los proveedores cloud. Utiliza algoritmos genéticos para realizar WSC con un módulo de negociaciones para los SLA. La composición es desplegada bajo un ambiente de SaaS, de manera general el usuario interactúa directamente con una interface SaaS que sirve de intermediario entre los proveedores de la nube y los servicios expuestos, esta estrategia de solución permite centralizar las peticiones de los usuarios, procesarlas tomando en cuenta atributos QoS y buscar correspondencia entre cual servicio y proveedor es el más adecuado para satisfacer el requerimiento dado por el usuario. El sistema es modelado en tres partes, la primera corresponde al procesamiento de la petición del usuario, la segundo a la búsqueda y selección del WS en los diferentes UDDI; finalmente, se procede a buscar los proveedores cloud según sus diferentes SLA. Con este mismo enfoque de trabajo sobre SaaS se encuentra la propuesta de Wakrime, *et al.* [73] donde ofrece un panorama sobre la WSC utilizando heurísticas basadas en *minimal unsatisfiability* que son utilizadas para el tratamiento de dominios donde predominan problemas de tipo NP. Se resalta de este trabajo la utilización de atributos QoS dinámicos, es decir, el problema es codificado constantemente si aparecen cambios en los requisitos de QoS , el sistema encuentra la composición mínima como una variante de insatisfacibilidad de una fórmula obtenida para el problema de composición modelado. La ventaja de este enfoque es el poder minimizar el costo global de la utilización de los WS necesarios en la actividad de composición, así como, controlar atributos generales (para el caso de la WSC) y atómicos para cada WS.

Un trabajo bajo restricciones QoS y SLA conscientes del contexto en ambientes cloud se desarrolla en [5]. La WSC se modela como un problema de optimización, cada WS se evalúa y selecciona tomando en cuenta restricciones QoS contenidas en los SLA, es utilizado un algoritmo que selecciona el mejor WS que no viole las restricciones prestablecidas por los requerimientos no funcionales. El aporte de esta investigación es un heurística basada en algoritmos genéticos que minimizan el costo computacional de búsqueda y selección de un WS, así mismo, ofrece un método de evaluación que detecta dentro de la WSC aquellos servicios que podrían eventualmente violar una condición de los SLA. La investigación resalta además el problema de la similitud de funcionalidades de los servicios junto con diferentes atributos QoS sobre un número creciente de clouds que conlleva a un problema de alta complejidad computacional.

En [60] es creado un sistema de selección dinámica de servicios en tiempo de ejecución basados en QoS, la composición es modelada como un problema de optimización y solucionado a través de programación entera, su sistema encuentra la solución óptima que satisfaga estos criterios por servicio y después el modelo se coloca a nivel externo donde se toma el comportamiento de los servicios en la actividad de composición. Basados en este trabajo también se encuentra

E3 [74], donde es desarrollado un marco de trabajo para optimizar la búsqueda de servicios que cumplen con un criterio de calidad, problema considerado NP cuando hay múltiples servicios que satisfacen un requerimiento no funcional, utiliza algoritmos genéticos multiobjetivo que reducen considerablemente el tiempo de búsqueda del servicio, los resultados son colocados en un conjunto de Pareto que almacenan soluciones óptimas que satisfacen múltiples condiciones de calidad.

La propuesta de [2] ofrece un panorama de categorización y clasificación de servicios a través del análisis y actualización de métricas de sus QoS,con procesos que evalúan la disponibilidad del servicio en términos de sus invocaciones. Su mecanismo denominado OAEQoS analiza el servicio con el valor medio de tres parámetros de calidad. Su estudio enfatiza la necesidad de mantener el criterio de calidad de disponibilidad actualizado para asegurar que en el momento de la selección de servicios, su resultado sea eficaz para instanciarlo en el flujo de una composición.

Finalmente, Zhou *et al.* [75], implementa un sistema para la selección de WS en ambientes cloud utilizando algoritmos Bayesianos. El trabajo establece ontologías para describir los servicios y sus proveedores junto con un esquema de relaciones que lleva a enriquecer un entorno semántico y permiten al sistema inferir sobre los posibles servicios que podría tener una composición. Con el modelamiento de la WSC se establecen métricas para identificar las similitudes entre WS a nivel de sus RNF.

La Tabla 3.3 presenta los anteriores enfoques centrándose en la tendencia moda , su estrategia de solución y los criterios QoS con los que trabajaron.

Utilizando está información, se detecta que las propuestas interactúan con motores de composición y un número reducido crean sus propias plataformas de despliegue para sus pruebas. Igualmente, los proyectos analizados presentan independencia en el momento de la entrada de datos a sus sistemas, unos basan sus entradas en dataset y otros registran de forma manual cada WS junto con sus proveedores y QoS de forma estática, razón que facilita la entrada de requerimientos al sistema y facilita el descubrimiento de servicios y el análisis de los modelos WSC . Por otra parte, realizan la documentación de los mecanismos de registro de WS, especificando su semántica y la forma como será operada por sus motores de búsqueda. Las búsquedas dinámicas se realizan por lo general en tiempo de diseño, algunos acercamientos a incorporación de WS en tiempo de ejecución son limitados al descubrimiento de un conjunto de servicios predeterminados en algún repositorio.

Tabla 3.3: Categorización de Enfoques de selección de WS

Enfoque	Moda	Estrategia	QoS
Ghobaei, et al. [64]	IA	Aplicación en multinubes, basado en algoritmo Cuckoo	Costo, Tiempo de Respuesta
Di, et al. [65]	IA	Sistema de búsqueda basado en ontologías-Se centra en búsquedas sintácticas del servicio y de sus atributos.	No especificado.

Continúa en la página siguiente.

Enfoque	Moda	Estrategia	QoS
Rangarajan, et al. [66]	ML,OP	Aprendizaje automático con búsquedas sobre los WSDL.	Accesibilidad, Rendimiento, Interoperabilidad y Tiempo de Respuesta
Chakravarthy, et al. [67]	PR	Clasificación a través de métricas de QoS usando árboles de decisiones.	Tiempo de respuesta, Rendimiento y Disponibilidad.
Sha, et al. [68]	IA,OP	Redes de petri - Cluster de servicios por similitud de QoS.	Criterios basados en un valor promedio de aceptación de un QoS. No especifica cuantos o cuáles son, se analiza el QoS general(ranking).
Wu, et al. [69]	IA,OP	Selección de servicios a través de relaciones de equivalencia, búsquedas a nivel sintáctico y de análisis de parámetros de entrada/-salida.	No especificado.
Qi, et al. [6]	ML,OP	Aprendizaje automático basado en técnicas de evolución diferencial.	Tiempo de respuesta, Disponibilidad y Fiabilidad
Sundareswaran, et al. [53]	OP	Proceso de selección basados en técnicas de indexación a través de en un broker que genera la categorización de servicios y proveedores .	Seguridad,costo y tiempo de respuesta.
Garg, et al. [70]	ML,OP	Aprendizaje automático basado en el proceso de jerarquía analítica.	Rendimiento, seguridad,costo y usabilidad
Gomes, et al. [71]	IA	Selección de servicios basado en técnicas dirigidas por modelo.	No especificado.
Bentaleb, et al. [72]	IA	Selección de servicios basado en algoritmos genéticos. Realiza un análisis de los WS con la información QoS dada en los UDDI.	Rendimiento,Fiabilidad, Precisión, Escalabilidad, Disponibilidad y Accesibilidad.
Wakrime, et al. [73]	IA,OP	Selección de servicios basado en un problema de satisfacibilidad booleana.	disponibilidad, costo y tiempo de respuesta.
Wang, et al. [5]	IA	Selección de servicios basado en algoritmos genéticos.	Tiempo de respuesta, disponibilidad, precio y reputación.

Continúa en la página siguiente.

Enfoque	Moda	Estrategia	QoS
Alrifai, et al. [60]	OP	Selección de servicios basado en programación lineal entera mixta.	Tiempo de respuesta,precio, disponibilidad y rendimiento.
Wada, et al. [74]	IA,OP	Selección de servicios basado en algoritmos genéticos multiobjetivo denominado E^3.	Rendimiento, latencia y costo.
Rathore, et al. [2]	OP	Clasificación y selección de servicios basados en la parametrización de sus criterios QoS. Ofrece un panorama de categorización de servicio.	Tempo de respuesta, fiabilidad y disponibilidad.
Zhou, et al. [75]	IA	Composición de servicios y selección usando análisis bayesiano.	Rendimiento, latencia y costo.

IA:Inteligencia artificial, ML: Machine learning, PR: Reconocimiento de patrones,OP:Optimización

Desde la perspectiva de las tendencias de moda, los estudios se engloban en su gran mayoría en estrategias computacionales derivadas de la inteligencia artificial, si bien está puede ser considerada una área que engloba a todas las estrategias, el estudio categoriza dos subáreas como machine learning y el reconocimiento de patrones. Un número considerable de enfoques consideran la composición y la selección de servicios como un problema de de optimización, enfatizando en la obtención de servicios eficaces y que hagan uso eficiente de sus recursos.

A nivel de los criterios QoS, el tiempo de respuesta , rendimiento y disponibilidad son las más utilizados a la hora de seleccionar WS, así mismo ,algunos enfoques utilizan otras métricas menos conocidas como la interoperabilidad, accesibilidad y el costo.

De igual manera, las investigaciones concuerdan en la utilización de criterios QoS que sean de fácil medición y obtención a través de los logs de los proveedores ya que estos datos pueden registrarse sin temor de colocar ruido en la información por la manipulación de los usuarios. Los criterios QoS no sólo son aplicados a los WS atómicos sino también al servicio compuesto, esto trae problemas inherentes de optimización ya que los WS seleccionados deben mejorar el objetivo buscado por esté servicio molecular. Algunos enfoques como [65, 69], especifican los criterios QoS como una única restricción para todos los servicios involucrados en la composición, esto limita considerablemente el proceso de selección en un contexto real ya que un servicio puede o no solicitar diferentes requerimientos no funcionales.

El trabajo de [2] exhibe una visión sencilla y útil de la categorización de un servicio en función de la valorización de criterios QoS con media aritmética de tres parámetros, según sus estudios el tiempo de repuesta,la fiabilidad y la disponibilidad son los parámetros de calidad más adecuados para procesos de selección. Está visión fue analizada y aporto un punto de comparación con respecto al trabajo en Ar_WSDS.

En función de lo analizado, la implementación y ejecución de una composición de servicios

debe tomar en cuenta la selección óptima de WS, entendiéndose esté último como el proceso en el cual se seleccionan los WS más adecuados a partir de unos criterios QoS para una modelo de composición en particular, de esté modo y debido al aumento exponencial de proveedores y servicios con similitud de RNF, se pueden encontrar conjuntos de servicios que puedan satisfacer esa composición.

Debe señalarse además, que las composiciones de servicios desde la perspectiva de la mayoría de estudios operan sobre un workflow que suponen un intercambio de mensajes entre servicios adyacentes. Es decir, debe existir una relación entre el servicio antecesor y sucesor modelado a través de un grafo acíclico. Esta forma de estudiar la composición ofrece un panorama sencillo para establecer los WS atómicos para cada nodo del modelo de composición. Esté hecho permite ver la gestión QoS para cada servicio atómico y molecular, estableciendo estrategias computacionales para solucionar los problemas de optimización local y global, respectivamente.

3.5. CONCLUSIONES

El descubrimiento y selección de WS se ve enfrentado al análisis de los diferentes criterios QoS con el que cuenta un servicio atómico. De tal forma, que el servicio compuesto no es más que una función que cumple con todas las restricciones dadas por esos RNFs. El reto de los enfoques reportados por literatura ha sido encontrar la forma más eficaz y eficiente de encontrar los servicios más adecuados para un modelo de composición.

Un número considerable de los enfoques estudiados utilizan como bancos de pruebas los datos de conjuntos abiertos que son alimentados después de una etapa de pre-procesamiento por parte de cada proveedor; en consecuencia de esto, el tratamiento de datos de manera online de los criterios QoS es aún incipiente y un desafío tecnológico.

CAPÍTULO 4

Reconocimiento de patrones para el descubrimiento y selección de servicios Web

El modelo de reconocimiento de patrones basado en el funcionamiento sistemático del cerebro para la selección de servicios es presentado en este capítulo. En él, se especifica cada una de las partes del sistema de reconocimiento y como son formalizados los servicios y la composición para ser representados en Ar_WSDS, la noción de patrón y señales son expuestos desde la perspectiva lógica-matemática. El proceso dará como resultados los servicios necesarios y suficientes que constituyen la composición donde cada servicio atómico cumplirá con sus restricciones QoS y como estos aportan a la calidad del servicio compuesto.

4.1. GENERALIDADES DEL RECONOCIMIENTO DE PATRONES

Computacionalmente, el reconocimiento de patrones es referido como un conjunto de algoritmos y técnicas cuyo fin es el de descubrir y reconocer una entidad (física o su representación en un modelo digital) a través de características elementales que permitan clasificarlos para un posterior proceso con las entidades reconocidas[76].

Los patrones de reconocimiento han sido ampliamente estudiados[77, 78], los trabajos apuntan a métodos de selección basados en características predefinidas sobre un conjunto de datos. El objetivo primordial del reconocimiento de patrones es su capacidad para encontrar y caracterizar información estructurada denominada patrones a partir de fuentes de información que presentan los datos de manera dispersa y que a través de diferentes estrategias permiten identificar tendencias con el reconocimiento de un descriptor(atributo) en específico[79]. Se definen dos estrategias de reconocimiento: supervisada y la no supervisada. La primera basa su reconocimiento en la presencia patrones estructurados, con esto se construyen la predicción de interés, en este punto es necesario una etapa de entrenamiento para determinar la pertenencia del patrón a una clase específica. De manera contraria, los no supervisados son capaces de

predecir comportamientos e interrelaciones entre los datos para determinar la función de pertenencia de una clase.

De manera general, los métodos basan su proceso en la eliminación de variables redundantes y en el reconocimiento rápido de una entidad con patrones activos o reglas de primer orden que esta debe cumplir para ser reconocida. Cada característica de la entidad corresponde a una variable que será analizada durante el proceso de reconocimiento, de está forma se realiza la categorización de la entidad. En los últimos años, las investigaciones han aportado mecanismos de reconocimiento de patrones que pueden analizar un conjunto considerable de variables y cuyo costo computacional resulta aceptable para un conglomerado de datos que en procesamientos básicos podrían tener un costo de orden NP[80].

Dentro de este marco de modelos reconocimiento se encuentra Ar2p[81] basado en el funcionamiento sistemático del cerebro. Ar2p, se construye a partir de la teoría de la mente para el reconocimiento de patrones (por sus siglas PRTM), en donde la memoria maneja una jerarquía de patrones y estos son percibidos a través de nuestros sentidos conceptualmente llamados "sensores", en el momento de que uno de nuestros sentidos se active, el patrón es reconocido.

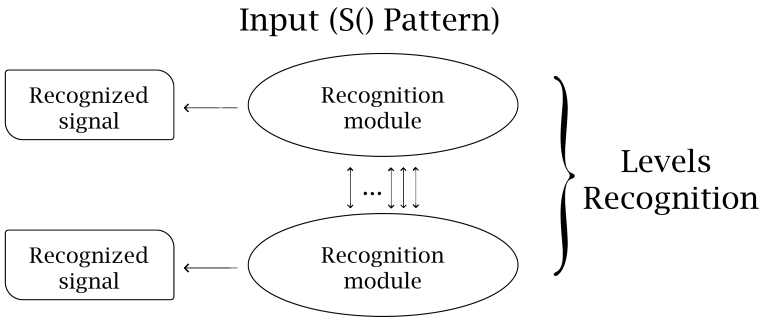


Figura 4.1: Modelo Ar2p.
Fuente: [81]

El modelo general de Ar2p es mostrado en la Figura 4.1, cada ovalo representa un módulo de reconocimiento, de esta forma se tienen varios niveles que pueden reconocer patrones atómicos(características con un sólo atributo/valor). El objetivo de cada módulo es reconocer su patrón correspondiente; es decir, una copia del patrón en el mundo real de tal forma que el reconocimiento puede darse en cualquier nivel[82]. Ar2p, define dos estrategias para reconocer un patrón, la primera por señales clave y la segunda por reconocimiento parcial de señales. El primero usa el peso de importancia de las señales de entrada identificadas como claves, y el segundo usa la presencia parcial o total de las señales, de esta forma, por ejemplo, un patrón puede ser reconocido por una sola señal clave o por la suma de las señales parciales, si y sólo si, estas pasan por un espacio de observación. La ventaja de la utilizar Ar2p es su nivel de uniformidad ya que ofrece que cada función de los módulos subyacentes de reconocimiento sean las mismas y este proceso es ejecutado recursivamente.

Ar_WSDS utiliza aprendizaje supervisado a través de lo propuesto en Ar2P para la creación

de una nueva versión de reconocimiento basados en QoS. En este sentido, el proceso a nivel general consistirá en la descomposición de cada parámetro QoS para ser analizado por módulos de reconocimiento creados dinámicamente hasta conseguir los WSs deseados para el modelo de composición dado por el desarrollador. A continuación, es formalizado el concepto de reconocimiento de componentes de software, composición y de servicio web que servirán como entrada al proceso de reconocimiento propuesto.

4.2. RECONOCIMIENTO DE PATRONES PARA COMPONENTES DE SOFTWARE

Los componentes de software constituyen parte importante en el desarrollo de aplicaciones como medio base para la convivencia e intercambio de datos entre los software construidos en diferentes lenguajes de programación. La noción de componente de software consiste en una *Unidad de Código* desplegado de forma independiente y que expone sus funciones a través de interfaces predefinidas por un lenguaje de intercambio de mensajes[83]. Dentro de este ecosistema, se encuentra los WS, definición presentada en la sección 2.4.

A partir de lo anterior expuesto, se propone la definición de **Reconocimiento de Patrones para Componentes de Software** como base para la formalización del proceso de reconocimiento, el concepto comprende:

"La capacidad para encontrar y caracterizar componentes de software a través de métricas y/o atributos definidos para su clasificación".

Del concepto precedente y adaptado a nuestro objeto de estudio, el reconocimiento de WS comprenderá el análisis de métricas QoS para seleccionar y clasificar servicios. Como resultado de la adopción de este concepto y de un sistema de reconocimiento de patrones, se presenta la Figura 4.2 donde el reconocimiento de componentes de software estará compuesto por las siguientes fases:

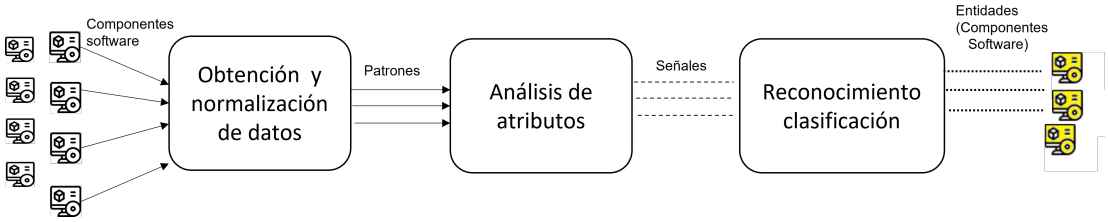


Figura 4.2: Reconocimiento de patrones para componentes de software.

1. Fase de obtención y normalización de datos, en ella se establece y depura la información del componente software a través de un lenguaje de definición propio de la unidad de código.
2. Fase obtención de características, su fin es establecer y determinar las características y/o atributos representados en métricas que van a servir para el reconocimiento del componente. Por lo general, estas características estarán enmarcadas dentro de los requerimientos no funcionales.

3. Fase de clasificación y reconocimiento, su objetivo es procesar las características a través de las diferentes estrategias y heurísticas computacionales para que las señales puedan ser reconocidas y clasificadas. Este proceso ofrecerá las entidades necesarias para que sistemas y/o aplicaciones realicen sus actividades de toma decisiones.

4.3. FORMALIZACIÓN DE LA COMPOSICIÓN Y DE SERVICIO WEB

La composición de servicios es definida como un modelo abstracto de composición (por sus siglas ACM) de WS que corresponde a un workflow secuencial como se muestra en la Figura.4.3 representado a través de un grafo acíclico de tamaño *n*, tal que existirán servicios de *1..n* que su intercambio de mensajes llevarán a la creación de un servicio compuesto o molecular. La Ecuación 4.1 representa el conjunto WS_T que contiene los servicios atómicos (nodos) necesarios para satisfacer la composición. Cada WS_i está compuesto por una 3-tupla: <I,O,[QoS]>, donde *I* son sus parámetros de entrada , *O* su interfaz de salida y *[QoS]* su matriz de parámetros de calidad deseados.

$$WS_T = \{WS_i \mid WS_i \in ACM\} \tag{4.1}$$

De esta forma existirá una matriz QoS_T donde un elemento $QoS_{i,j}$ es un valor de un atributo de calidad *j* para un WS en la posición *i* dentro del ACM. Es decir, el WS_1 tiene un valor de criterio de calidad $QoS_{1,0}$ que puede ser por ejemplo, latencia y así sucesivamente para cada servicio. Por consiguiente, cada WS tendrá asociado una matriz de parámetros QoS. Este conjunto es denotado por un arreglo de matrices QoS_T.

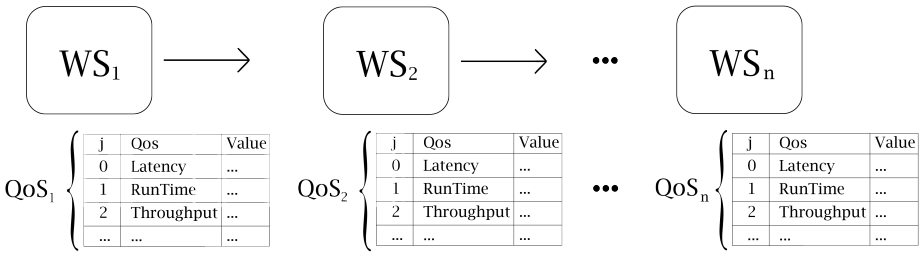


Figura 4.3: Modelo Abstracto de composición.

Cada paso por el ACM requiere del consumo de un WS_i. El objetivo de la composición se alcanza cuando todos los WS_i están disponibles y correlacionados en el modelo, de tal forma que la ejecución es basada en el recorrido ordenado de cada nodo al cumplir con la funcionalidad adyacente. Si bien pueden existir otros tipos de modelos de workflow, estos podrán ser transformados en modelos secuenciales, proceso estudiado en [84]. Cada WS debe cumplir con una condición de ensamblaje dado por su modelo de composición, esto corresponde al llamado e invocación de un WS antecesor con su candidato predecesor.

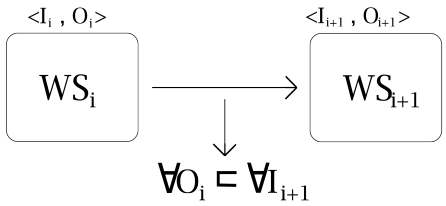


Figura 4.4: Estructura del ensamble de WS en una composición.

Cada WS debe cumplir con una condición de ensamblaje, denominado CE, dado por su modelo de composición, esto corresponde al llamado e invocación de un WS antecesor con su candidato predecesor. La Figura 4.4, presenta la relación transitiva entre los parámetros I y O, de nodos adyacentes. El desarrollador se encargará de realizar la correspondiente validación para el cumplimiento del RF de su composición.

Los WS de entrada para el sistema son denotados por el conjunto WS_g, donde WS_{g_i} es un servicio web atómico a ser reconocidos por el sistema. Un WS candidato denotado como WS_c es un servicio que ha sido descubierto y seleccionado a partir de los WS_g y es considerado una posible solución para un nodo del ACM. La combinación de los WS_c crearan las rutas de solución o los diferentes subconjuntos que satisfacen el ACM y las condiciones QoS dadas por el desarrollador.

Finalmente, con la explicación antecedente, se denota un ACM como una estructura formal del tipo:

$$ACM =< WS_T, CE, \xi >$$
$$\xi : CE \rightarrow WS_T$$

Donde WS_T son un conjunto no vacío que representa los servicios o nodos del grafo ACM, CE su condición de ensamble y ξ [1] son aplicaciones de CE en WS_T.

4.4. ARQUITECTURA AR_WSDS

Ar_WSDS fue concebido desde una arquitectura modular que ofrece las ventajas de dividir el sistema en funcionalidades que son fácilmente desarrolladas por cualquier sistema software. La idea de modularidad es la capacidad de que cada una de sus partes pueda ser implementada de manera independiente[85]. La estructura de Ar_WSDS es presentada en la Figura 4.5 en ocho partes funcionales.

[1]Computacionalmente, ξ representa los llamados o invocaciones de los servicios en el workflow definido por el desarrollador.

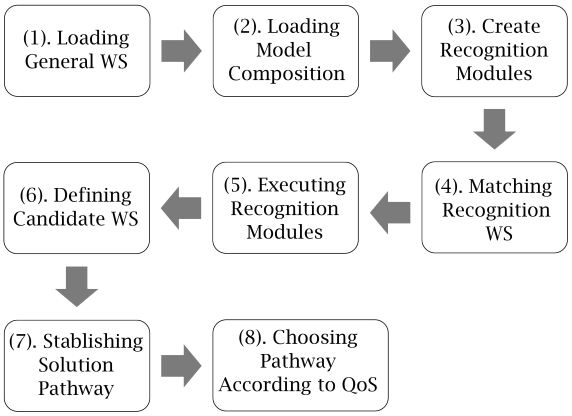


Figura 4.5: Estructura de Ar_WSDS

Parte 1. Corresponde a la entrada del sistema, cuya función es la de leer los datos, normalizar y crear el conjunto WS_g. Estos datos son registrados a nivel atómico y el criterio de calidad de se desea alcanzar en la composición.

Parte 2. Carga del modelo de composición que define los requerimientos de sintaxis del servicio y los parámetros QoS a nivel atómico y molecular.

Parte 3. Creación dinámica de los módulos de reconocimiento que se ajustan al ACM, a través de las dos estrategias establecidas en Ar_WSDS (analizadas en la Sección 4.5).

Parte 4. Proceso de búsqueda sintáctica por nombre de servicio, su fin es el de reducir el espacio de búsqueda en el conjunto WS_g.

Parte 5. Proceso de ejecución de los módulos de reconocimiento, su fin es el de realizar la preselección de los WS_c con los servicios definidos en el paso anterior.

Parte 6. Proceso de definición de candidatos, su fin es el de tomar los WS_c y filtrar los servicios que aporten a la función de calidad definida por el desarrollador para su composición.

Parte 7. Se establecen los conjuntos de WS_c que satisfagan cada nodo del ACM.

Parte 8. Proceso de selección de rutas de solución, cuyo fin es depurar el conjunto de soluciones y escoger aquellos que conjuntos que estén por encima de la métrica de calidad definida para el servicio compuesto.

A continuación se presentan las definiciones del sistema y las anotaciones semánticas utilizadas en Ar_WSDS como una serie de componentes que identifican el sistema dentro del entorno PRTM.

Definición 1 *Un patrón (ρ) es una entidad representada por un WS_{g_i} a ser reconocido a través de los parámetros dentro del conjunto QoS_T. A partir de la descomposición de ρ se crea el conjunto de señales atómicas denominadas S().*

S() contiene información sobre cada WS_{g_i}, y los módulos de reconocimiento clasifican el servicio como candidato si cumple con las restricciones QoS.

Definición 2 *Un WS óptimo es un servicio cuyo procesamiento de cada uno de sus criterios QoS*

está en un umbral definido por las restricciones QoS definidas en el ACM.

Definición 3 *Una ruta de solución (por sus siglas en inglés SP) es un subconjunto de WS óptimos. El sistema reconoce 0 o n SP. Si su valor es 0, significa que algún servicio no cumplió con su respectivo parámetro de QoS; por lo tanto, un nodo ACM no tendrá ningún servicio asociado, violando la aplicación definida en ξ.*

La Figura 4.6, presenta un ejemplo del sistema Ar_WSDS, en el que las entradas son las especificaciones WS_g y el ACM. A continuación, los módulos de reconocimiento descubren y seleccionan los WS más adecuado para cada nodo del modelo.

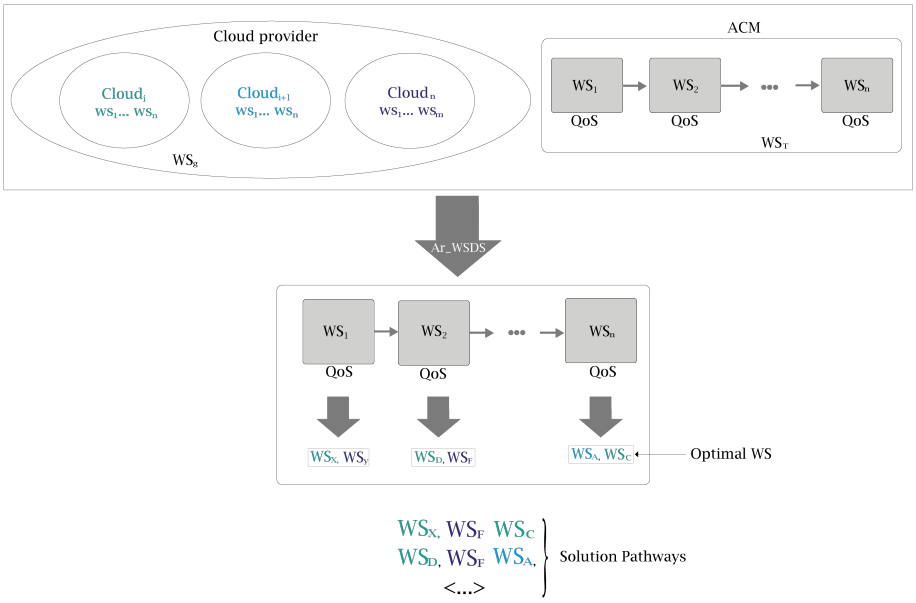


Figura 4.6: Ejemplo de reconocimiento a través de Ar_WSDS.

Definición 4 *Un módulo de reconocimiento, representado como (Γ) es un conjunto de componentes de software capaces de reconocer un patrón y sus señales.*

$$\Gamma = \{\Gamma_{i,j} \mid \Gamma_{i,j} \in QoS_{i,j}\} \tag{4.2}$$

Por modularidad y arquitectura de Ar_WSDS estos módulos son dinámicos, es decir, adaptables automáticamente a las parámetros de cada WS_i del modelo de composición. La Ecuación 4.2 formaliza los módulos de reconocimiento, donde:

1. $\Gamma_{i,j}$ corresponde a un módulo de reconocimiento para el servicio i de su parámetro QoS en la posición j.
2. $QoS_{i,j}$ corresponde a un valor de parámetro QoS en la posición j del servicio i.

Definición 5 *Ranking de un WS representado como (φ) es un valor del grado o peso de importancia que tendrá un WS_c cuando es reconocido por los módulos. (Γ).*

$\varphi_{i,j}$ tendrá un valor de *1* cuando el valor para $QoS_{i,j}$ del WS_{g_i} está categorizada como el más óptimo y un valor de *n* para servicios que se encuentran alejados de los parámetros QoS establecidos en la composición de cada nodo. Su proceso es explicado en detalle en la sección 4.6.

Definición 6 *La función matching de reconocimiento, representada como $M()$ evalúa si el nombre del servicio descrito en el conjunto WS_T posee correspondencia sintáctica con algún servicio del conjunto WS_g, en cuyo caso su resultado es true, o false en caso contrario.*

La Ecuación 4.3, presenta la función M() cuyo objetivo es minimizar el espacio de búsqueda entre los WS_g y crear los candidatos primarios (elementos detonados como x). Estos serán WS a ser tratados por los módulos de reconocimiento que analizarán cada valor de sus QoS.

$$\exists_x \left\{ x \mid x \in WS_c \leftrightarrow M\left(WS_g\right) = true \right\}. \tag{4.3}$$

Definición 7 *Los umbrales de calidad, representados como (Δ), corresponden a un valor discreto que deben cumplir la composición(servicio molecular) y cada WS_{g_i} (servicio atómico) para ser WS_c.*

Según la naturaleza,los QoS sus valores óptimos estarán dado en máximos o mínimos, la caracterización de los atributos son descritos en [86] y analizados desde la perspectiva de los datasets en [87]. Los Δ son de tres tipos:

1. ΔACM es un valor(porcentaje) que representa la condición de calidad que debe cumplir el servicio molecular(compuesto), a través del proceso de analizar los WS_c y su φ respectivo, para ser considerado dentro de una ruta de solución.
2. ΔKS umbral clave es un valor de un parámetro de calidad para un WS_{g_i} que debe estar en el intervalo de aceptación de su $QoS_{i,j}$ respectivo. Como se ha mencionado depende de la naturaleza del parámetro y puede ser expresado como mínimos o máximos. El valor ΔKS es obligatorio su cumplimiento.
3. ΔPS umbral parcial, es un valor de parámetro de calidad para un WS_{g_i} que está por encima o por debajo del promedio de sus QoS del mismo tipo. Estos valores pueden estar por debajo de su condición de calidad. El valor de φ_{ij} es sometido a una función de penalización denominada $\beta(\varphi)$ que presenta la categorización del valor $QoS_{i,j}$ y cuyos valores de rankings son calculados a partir de la diferencia entre el mejor y el servicio más alejado de la condición de calidad.

De esta forma, para que un WS_{g_i} sea considerado candidato deberá cumplir que cada $QoS_{i,j}$ este en los umbrales definidos en ΔKS o bien en los ΔPS.

Definición 8 *Tipo de reconocimiento es un valor (0,1) asociado a un $QoS_{i,j}$ para que utilice un Δ en particular, si el valor es 1 utilizará ΔKS o 0 para ΔPS . Ningún valor Δ significa que el parámetro QoS no será considerado por los módulos de reconocimiento.*

A partir de lo presentado en la Figura 4.1 se extrae un servicio y se presenta un ejemplo de WS con sus parámetros QoS, la Figura 4.7 describe los valores y tipo de Δ; latencia y runtime utilizarán reconocimiento usando ΔKS y throughput usará ΔPS.

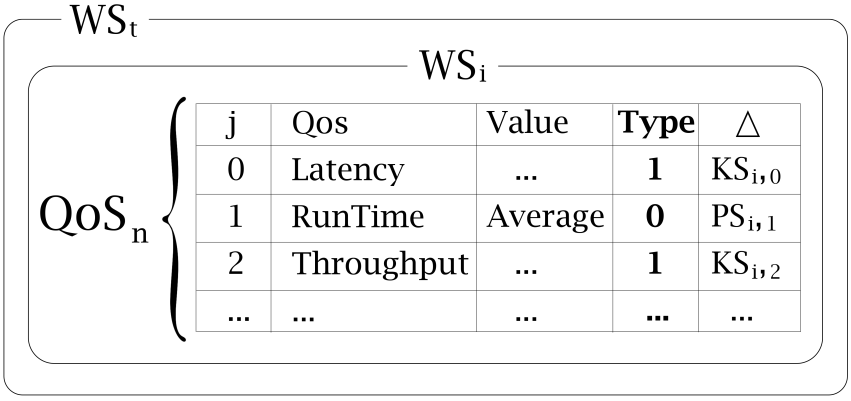


Figura 4.7: Especificación de un WS con su tipo de reconocimiento.

4.5. ESTRATEGIAS DE RECONOCIMIENTO

El sistema opera con dos estrategias para el reconocimiento del conjunto de señales a partir de los umbrales ΔKS y ΔPS, donde se descomponen en señales atómicas S() correspondientes a las restricciones $QoS_{i,j}$, formalizadas en las siguientes definiciones.

Definición 9 *Módulo de reconocimiento por señales claves, es un proceso de reconocimiento que se activa cuando un módulo Γ_{ij} analiza para cada WS_{g_i} su ΔKS y obtiene un valor $\varphi_{i,j} \in \{1..n\}$. Si $\varphi_{i,j} = 0$ la señal $S()$ no es activada y el elemento analizado del conjunto WS_g es descartado de WS_c.*

La Ecuación. (4.4) presenta el modelo para el reconocimiento de un patrón (WS_i) cuando la suma de todos sus rankings generados de la evaluación en Γ_{ij} en sus $QoS_{i,j}$ sean mayores o iguales a su ΔKS respectivo y $\varphi_{i,j} > 0$. El numerador de la ecuación en la sumatoria, determina la proporcionalidad exacta del QoS para que el servicio sea reconocido. El denominador determina la contribución que tiene ese parámetro para alcanzar su ΔKS.

$$KS\_Recognizer\left(WS_{g_i}\right) = \sum_{j=1}^{|QoS_i|} \left(\frac{\frac{100}{|QoS_i|}}{\varphi_{i,j}} \right) \geq \Delta KS_{i,j} \rightarrow WS_{g_i} \in WS_c \tag{4.4}$$

Definición 10 *Módulo de reconocimiento por señales parciales, es un proceso de reconocimiento que se activa cuando un módulo Γ_{ij} analiza para cada WS_{g_i} su ΔPS , que se obtiene como la media de los QoS de la misma naturaleza de cada servicio atómico y obtiene un valor $\varphi_{i,j} \in \{1..n\}$. Si $\varphi_{i,j} = 0$, la señal $S()$ no es activada y el elemento analizado del conjunto WS_g es descartado del conjunto del WS_c.*

Una señal $S_{i,j}$ activa un módulo de reconocimiento por señales parciales Γ_{ij} cuando la suma de todos sus rankings generados a partir de Γ_{ij} sobre la función de penalización $\beta(\varphi)$ sean mayores o iguales a su ΔPS respectivo, formalizada en la Ecuación (4.5).

$$PS\_Recognizer\left(WS_{g_i}\right) = \sum_{j=1}^{|QoS_i|}\left(\frac{\frac{100}{|QoS_i|}}{\beta\left(\varphi_{i,j}\right)}\right) \geq \Delta PS_{i,j} \rightarrow WS_{g_i} \in WS_c \tag{4.5}$$

La función de penalización determina un valor cercano a 1 cuando el parámetro del QoS para WS_g está sobre la media dependiendo su naturaleza(máximos o mínimos), o un valor de ranking mucho mayor cuando se aleje de este parámetro de comparación. Por lo tanto, ofrece la posibilidad de que el sistema clasifique los WS del más adecuado hasta el de menor aporte de calidad, en términos de alcanzar los respectivos ΔPS.

La búsqueda y selección de los WS_c se realizan a través de los módulos de reconocimiento. El sistema inicia cuando las señales S() que contienen la información del WS atómico son reconocidos por la función M(). Su objetivo es el de verificar sintácticamente si el WS corresponde con el requerimiento de identificación solicitada por el ACM. Las señales son analizadas según su naturaleza y su tipo de reconocimiento *KS* o *PS*, se realiza el cálculo de los rankings generados por los diferentes módulos de reconocimiento y se establece la lista de los WS_c que garanticen ΔACM.

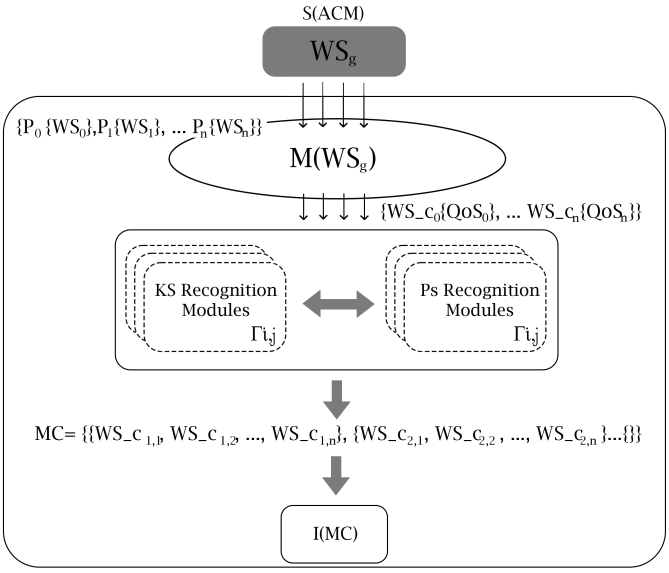


Figura 4.8: Sistema de reconocimiento Ar_WSDS.

La Figura 4.8 presenta el sistema de reconocimiento Ar_WSDS, que recibe como entrada su modelo de composición y los servicios web que deben clasificarse desde un repositorio. La función $M()$ realiza la búsqueda primaria y crea los candidatos iniciales y los envía al módulo de reconocimiento correspondiente, a partir del reconocimiento seleccionado que analizarán la naturaleza de cada QoS y sus valores. El sistema procesa y genera los candidatos para crear los SP que se derivan de aquellos que logran alcanzar la especificación de calidad dada en ΔACM.

A partir de a Figura 4.8, es desarrollado el Algoritmo 4.1, su funcionamiento es descrito a continuación :

1. Leer los datos del conjunto WS_g
2. Leer los datos del ACM. Este procesos establece la especificación de los servicios atómicos y cada atributo QoS (se registra los Δ respectivos). La información es almacenada dinámicamente en el vector WS_T.
3. Declaración y creación de la matriz dispersa dinámica $MC[][]$ [2], la cantidad de filas dependerá de la cardinalidad del vector WS_T.
4. Creación de los candidatos primarios, a partir del análisis sintáctico realizado por la función $M()$, su resultado establece los primeros elementos de WS_c. Este paso permite reducir el espacio de búsqueda y sólo centrarse en la operacionalización de aquellos WS_{g_i} que puedan pertenecer a una SP.
5. Para cada WS_i, WS_T es descompuesto en sub-patrones $\rho(WS_i)$. Ellos generan los correspondientes módulos reconocedores Γ, los cuales son los responsables del proceso de reconocimiento determinado por ΔKS o bien por un ΔPS según los requerimientos del desarrollador. Por cada WS_c, su módulo Γ procesa, si el reconocido, procede a calcular su ranking (ρ).
6. La función de reconocimiento según sea el caso son formalizadas en las Ecuaciones 4.4 y 4.5. Cuando el reconocimiento es válido, el elemento WS_c es almacenado en la matriz MC (en su columna respectiva). La Ecuación 4.6, define el formalmente el elemento reconocido, donde n es el número de $WS \in WS_T$ y m es el número de servicios que satisfacen los nodos en ACM, al ser una matriz dispersa este valor puede ser diferente para cada fila. El elemento x definido en la ecuación será un WS reconocido para ser utilizado en las rutas de solución.

$$MC_{nxm} = \exists_x \left\{ x \mid x \in \begin{bmatrix} WS_{c_{1,1}} & \dots & WS_{c_{1,m}} \\ WS_{c_{2,1}} & \dots & WS_{c_{2,m}} \\ \dots & \dots & \dots \\ WS_{c_{n,1}} & \dots & WS_{c_{n,m}} \end{bmatrix} \leftrightarrow (KS(x) \bigoplus PS(x)) \right\}. \tag{4.6}$$

7. El proceso no es válido, si no existe un WS_c para algún WS_i; puesto que, no existiría servicios para un nodo en particular de la composición.
8. La función $I(M_c)$ se encarga de integrar las rutas de solución establecidas en M, proceso realizado con la agrupación de los servicios candidatos por cada WS del modelo y los selecciona según la condición ΔWSC que satisfacen el modelo de composición.

4.6. RANKING O CATEGORIZACIÓN DE SERVICIOS WEB

Categorizar o dar un ranking de un servicio web consiste en otorgar una posición a través de un valor discreto que represente el grado de importancia del WS como candidato en un nodo

[2]Computacionalmente la creación de una matriz dispersa, sólo se hace necesario la especificación de sus filas como estructura de datos estática, está particularidad permite que sus columnas sean creadas en tiempo de ejecución, en el caso particular de Ar_WSDS dependerá de lo encontrado en los WS_c para cada nodo del ACM.

Algoritmo 4.1: Algoritmo de reconocimiento de patrones de Servicios Web.

```
Input: ACM,WS_g[]
Result: MC[][]
WS_T[] = load(ACM);
WS_CT[] = M(WS_g);
MC[0..|WS_T| - 1] [ ] be_a_new_matrix;
i = 0;
for WS_i ∈ W_T do
    Γ = ρ(WS_i);
    j = 0;
    for WS_c ∈ W_CT do
        if Γ(WS_c) and Γ ∈KS then
            isC = KS_Recognizer(WS_c);
        else
            isC = PS_Recognizer(WS_c);
        end
        if isC = true then
            MC[i][j] = WS_c;
            j = j + 1;
        else
            delete WS_c in WS_CT;
        end
        if M[i] = ∅ then
            Error no candidate services;
            exit;
        end
        i = i + 1;
    end
    Create solution pathways based on MC;
    I(MC);
end
```

del ACM. Ar_WSDS opera esté concepto a través de lo establecido en la función φ y presentado en la definición 5.

La importancia de la creación del ranking radica en el hecho de ofrecer un mecanismo para seleccionar eficazmente un servicio debido a la cantidad de WS con similitud de funcionalidades y criterios QoS, problema descrito en la sección 3.1.

Ar_WSDS clasifica los servicios disponibles en función del análisis de sus criterios QoS durante su proceso de reconocimiento de sub-patrones. No obstante, es importante resaltar que el análisis de QoS se basa en repositorios que a priori han realizado la tarea de especificar,

almacenar y actualizar los parámetros QoS de sus servicios.

El mecanismos de reconocimiento utiliza dos niveles de categorización o de ranking, descritos a continuación:

4.6.1. Ranking de un QoS para un SW

El mecanismo de categorización de un QoS para un SW , utiliza como base la algoritmia derivada de la utilización de estructuras de datos dinámicas en memoria denominadas *Árboles binarios de búsqueda* o por sus siglas ABB.

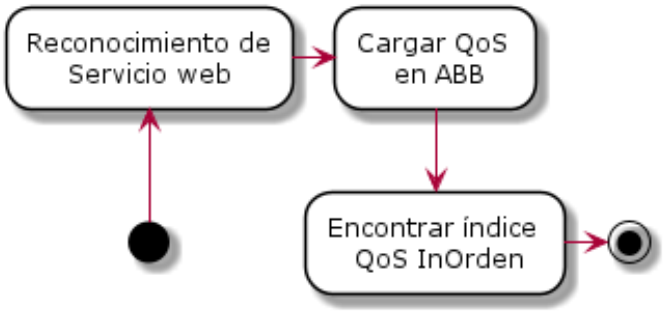


Figura 4.9: Flujo de trabajo del proceso de ranking de un QoS.

Dentro de un ABB, las operaciones de inserción y búsqueda tienen un costo computacional de $log_2 n$ [88]. Cada nodo del ABB corresponderá a un arreglo de WS_{ci}, donde i es el índice del WS del ACM. La Figura 4.9, presenta el proceso de ranking para cada WS y sus criterios QoS, en el se describen las siguientes acciones:

1. Proceso de reconocer un SW a través de las estrategias de reconocimiento por señales claves o parciales. Esté proceso asegura que sólo serán categorizados los servicios que van hacer parte de las rutas de solución.
2. Para cada QoS de un WS_c es insertado en el ABB a partir de su factor de inserción [3], creado a partir del análisis del ΔKS y ΔPS establecidos por el desarrollador. Las Ecuaciones 4.8 y 4.7 presentan el cálculo del factor para valores de QoS que necesitan ser maximizados o minimizados según sea su naturaleza, denominados IFMAX e IFMIN, respectivamente. Δ hará referencia a un umbral KS o PS.

$$IFMAX(QoS_{i,j}) = \Delta_{i,j} - QoS_{i,j} \tag{4.7}$$

$$IFMIN(QoS_{i,j}) = QoS_{i,j} - \Delta_{i,j} \tag{4.8}$$

[3]En un ABB el factor de inserción define el orden de inserción de los datos, donde el valor del factor de inserción de la raíz es menor o igual a su sub-árbol izquierdo y mayor a su sub-árbol derecho.

3. El índice en el recorrido inorden corresponde a la posición que ocupa el valor $QoS_{i,j}$ dentro del ABB. Este recorrido en inorden dará una secuencia ordenada ascendentemente,donde el primer valor comenzará con "1"hasta "n"denominado φ'. Los φ' pueden tener valores iguales, está característica se da cuando los WS_c tienen el mismo valor de $QoS_{i,j}$. La Ecuación 4.9 formaliza la función para obtener el índice de un valor QoS.

$$\varphi(Qos_{i,j})' = get\_index\_inorder(QoS_{i,j}) \tag{4.9}$$

	WSc_1	WSc_2	WSc_3	WSc_4
	$QoS_{i,j+1}$	$QoS_{i,j+2}$	$QoS_{i,j+3}$	$QoS_{i,j+4}$
Index_Inorder:	**1**	**1**	**2**	**3**

Figura 4.10: Ejemplo de cálculo de φ'.

La Figura 4.10 muestra un ejemplo de cálculo de un φ', donde existen cuatro WS candidatos. Es analizada una condición QoS, para este caso los candidatos 1 y 2 se encuentran de *primeros*, tienen el mismo valor de categorización (iguales valores QoS); el candidato 3 tiene un valor de categorización que implica que está de *segundo*; y finalmente, el candidato 4 que se encuentra de *tercero*, estas denominaciones otorgan la descripción del ranking.

La Figura 4.11 presenta un prototipo de un ABB con seis rankings. El recorrido inorden[4] del ABB da el orden de visita de cada nodo que corresponde con su valor de categorización. En él se puede notar que existen WS con el mismo valor de ranking y otros más alejados al valor de su restricción en su correspondiente Δ. Computacionalmente, el conjunto WS_c más eficaz para el procesamiento de un valor QoS en particular, se encontrarán en los nodos más izquierdos.

Este modelado taxonómico permite operar con varios procesos de consulta, por ejemplo, ¿Cuantos WS con una restricción QoS cumplen con una métrica de calidad?, ¿Cuáles son los WS más eficaces sobre una restricción QoS?, ¿Cuáles WS pueden mejorar sus condiciones de calidad más fácilmente a partir de un RNF dado por el desarrollador?; entre otras preguntas que pueden ser inferidas a través de la representación arbórea de los rankings.

El mecanismo de ranking desarrollado en Ar_WSDS clasifica los WS que ya fueron reconocidos por las estrategias KS Y PS, y sugiere los servicios que alcancen las diferentes condiciones de calidad y restricciones dadas por el desarrollador. Está forma de concebir el sistema otorga la posibilidad de encontrar más soluciones sobre el límite colocado en la especificación de los umbrales Δ.

[4]El proceso de recorrer en inorden o en espejo un árbol binario de búsqueda, comprende la visita de su sub-árbol izquierdo, raíz-sub-árbol y visitar sub-árbol-derecho. Comportamiento que genera una secuencia ordenada por su factor de inserción.

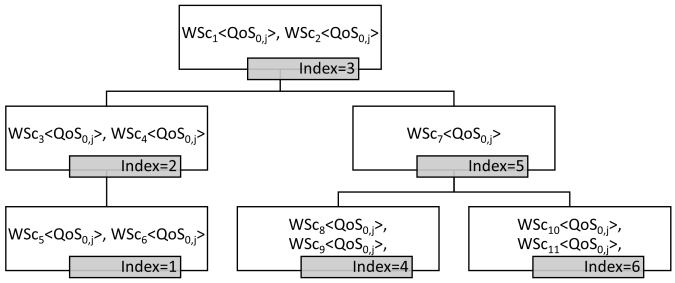


Figura 4.11: Prototipo de ABB del mecanismo de ranking.

4.6.2. Ranking de un SW

La caracterización del ranking de un WS, denotado por la función φ, es realizado a partir de lo obtenido por cada categorización del criterio QoS, donde se establece un sólo valor que permite procesar estos resultados y darles una ponderación que define el puesto del WS dentro de su conjunto de candidatos.

El valor φ clasifica un WS utilizando un factor de ponderación mediante los valores $\varphi'_{i,j}$. La ponderación utiliza como peso del servicio más óptimo el valor de *1*, por lo tanto se realiza una relación de proporcionalidad para determinar el grado de aporte que cada φ' da a la categorización final del servicio. Al igual que el proceso de ranking para un QoS, se utiliza la misma estrategia computacional para la carga de datos sobre un ABB.

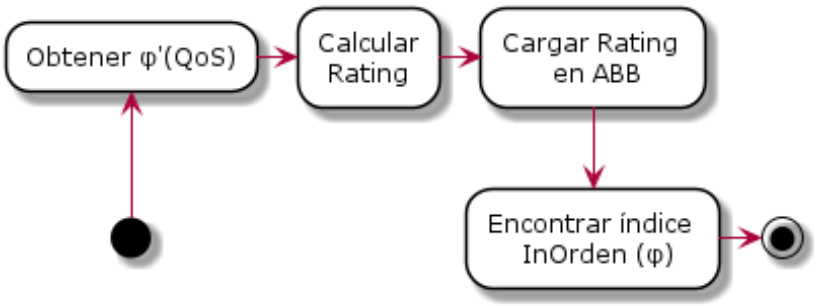


Figura 4.12: Flujo de trabajo del proceso de ranking de un servicio web.

El flujo de trabajo de este proceso es presentado en la Figura 4.12 y se describe a continuación:

1. Obtener el valor para cada WS de su $\varphi'_{i,j}$.
2. Cargar los WS a partir del factor de ponderación (rating), definido en la Ecuación 4.10. La clave de inserción utiliza esté factor de ponderación, en consecuencia de la estrategia computacional que utiliza el ABB, se realiza un producto por un valor inverso(-1) para que datos de mayor valor queden taxonómicamente en los sub-árboles izquierdos.

$$Rating(WS_{ci}) = \left\{ \sum_{j=0}^{|WS_c|} \frac{1}{\varphi'_{i,j}} \right\} * (-1) \tag{4.10}$$

3. Obtener el índice en el recorrido inorden, esto establece el orden que un WS ocupa dentro de todos sus WS_c. La Ecuación 4.11 formaliza la función φ que es utilizada en las estrategias de reconocimiento KS y PS , respectivamente.

$$\varphi(WS_{ci}) = get\_index\_inorder(WS_{ci}) \tag{4.11}$$

4.7. EJEMPLIFICACIÓN GENERAL DEL MODELO

Ar_WSDS ofrece un servicio de descubrimiento y selección de WS a partir de dos parámetros de entrada analizados en la sección anterior. El primero corresponde al ACM y el segundo a los servicios registrados en un repositorio.

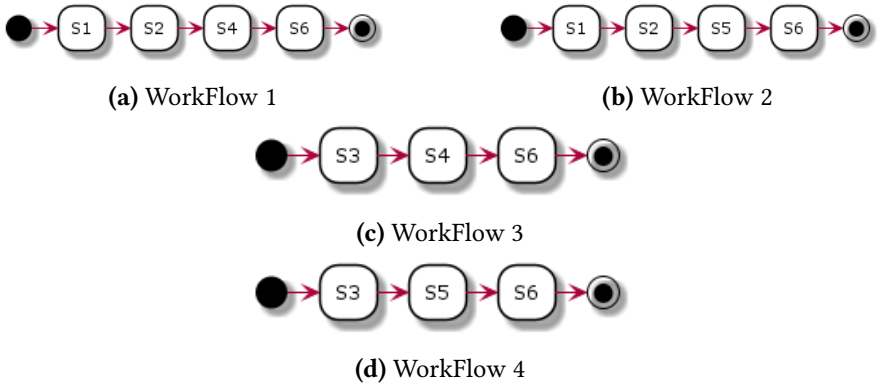
Ar_WSDS opera con workflows secuenciales, aunque se pueden encontrar flujos de trabajo paralelos, el sistema se encarga de seleccionar todos los servicios que aparecen en la composición y no evalúa el proceso de ejecución de la composición. Ar_WSDS opera únicamente en la fase de descubrimiento y selección. Los umbrales definidos son de dos tipos, el primero relativo al umbral de calidad de la composición y el segundo relativo al conjunto de criterios de QoS que debe cumplir cada uno de los servicios atómicos, es decir, el sistema selecciona aquellos servicios que contribuyen al objetivo de calidad fijado por el Desarrollador.

La representación serializada del grafo sobre un modelo de composición, ofrece las ventajas de adaptabilidad y fiabilidad para una composición de WS debido que se pueden identificar de manera clara los problemas críticos a la hora de la invocación de servicios y que pueden observarse desde la primera etapa del diseño de la composición en el cual es necesario la identificación de todo el dominio de servicios como la de sus restricciones de uso.

Un ejemplo de esto, se puede tomar a partir de la Figura 2.8 y a través de lo propuesto en [84], donde se pueden obtener varios flujos con el diseño de la composición dado por el desarrollador y dando como resultado los workflows alternos mostrados en la Figura 4.13 y analizados a con las convenciones de la figura para los elementos $\oplus$ y $\otimes$, respectivamente. En esté caso, existirá un flujo alterno entre S_1 y S_3, otro para S_4 o bien S_5, y como WS de obligatorio operación el S_6.

Ar_WSDS ofrece los mecanismos para que la búsqueda y selección de WS se realiza a partir de las preferencias del desarrollador,no operando en la definición de nuevos flujos a partir de un ACM dado, la entrada al sistema será el workflow que sea seleccionado para su operacionalización, así el podrá escoger como entrada los flujos 4.13a, 4.13b, 4.13c o bien 4.13d.

Este tipo de solución se planteó en base al hecho de que no todos los servicios tendrán los mismos criterios de calidad y la personalización de la búsqueda por servicio es un factor clave en la composición. La mayoría de las soluciones reportadas en la literatura utilizan un QoS



(a) WorkFlow 1

(b) WorkFlow 2

(c) WorkFlow 3

(d) WorkFlow 4

Figura 4.13: Ejemplo de conjuntos de Workflows para una composición

global y cómo los servicios contribuyen a lograr el objetivo, el sistema propuesto evalúa cada servicio a nivel de personalización de su QoS.

Ar_WSDS considera el ACM en una estructura en serie; sin embargo, al basarse de módulos de reconocimiento dinámicos, se pueden añadir componentes que analicen los criterios QoS para el servicio compuesto y establecer heurísticas para diferentes tipos de flujos de trabajo con el fin de redefinir el proceso de selección, esto permitiría obtener los servicios mínimos que satisfagan un requisito funcional.

El QoS global se toma como un valor de calidad que debe alcanzar cada SP. Este tipo de adaptación permite mejorar el seguimiento de las propiedades no funcionales de los servicios atómicos, y esta característica se denota en las composiciones dinámicas que requieren cambios estructurales en la funcionalidad de un servicio compuesto sin alterar el requisito funcional para el que fue creado.

La Figura 4.14 presenta un modelo de ejemplo donde se cuenta con un workflow con cuatro WS y estos son reconocidos usando Ar_WSDS. En la solución se puede apreciar que existen dos servicios que satisfacen a WS_1, un servicio WS_2 , dos para WS_3 y, finalmente dos para WS_4. Estos WS estarán almacenados en la matriz M de candidatos. A través de la función $I(MC)$ se crean las rutas de solución que sean mayores a iguales al valor definido en Δ del servicio molecular. Esta condición de calidad es inversamente proporcional a la cantidad de tuplas de las rutas de solución.

La Tabla 4.1, presenta la especificación de dos WS que hacen parte del conjunto WS_g con sus respectivos valores QoS, para efectos del ejemplo, la naturaleza de estos QoS pertenecen a criterios que deben ser minimizados. La Tabla 4.2, muestra los requisitos de la composición definiendo un $\Delta ACM = 80\,\%$, acá se puede notar que las señales de la 1 a la 4 van a ser reconocidas usando señales claves(KS) y de la 5 a la 9 parciales(PS). Esto quiere decir, que si el sistema no es capaz de encontrar ningún WS por señales claves, utiliza el reconocimiento parcial.

Tomando en cuenta la función M() y las Ecuaciones 4.4 y 4.5, Ar_WSDS trabaja de la siguiente

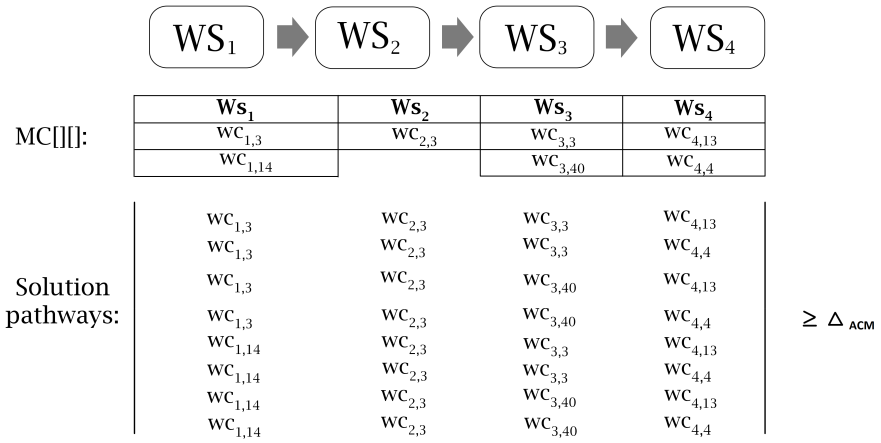


Figura 4.14: Ejemplo genérico de reconocimiento usando Ar_WSDS.

Tabla 4.1: Ejemplo de un Servicio Web y sus criterios QoS

WS	QoS_1	QoS_2	QoS_3	QoS_4	QoS_5	QoS_6	QoS_7	QoS_8	QoS_9
1	266.83	35	0.9	37	60	89	69	15.91	7
2	261	34	0.9	37	60	89	69	13.67	12

forma:

1. El sistema reconoce con la función M() dos posibles candidatos WS_1 y WS_2.
2. Se crea la matriz QoS:

$$QoS = \begin{bmatrix} 266{,}83 & 35 & 0{,}9 & 37 & 60 & 89 & 69 & 15{,}91 & 7 \\ 261 & 34 & 0{,}9 & 37 & 60 & 89 & 69 & 13{,}67 & 12 \end{bmatrix}$$

3. Se crean los módulos de reconocimiento Γ. La Tabla 4.3 presenta la evaluación de los criterios QoS. Las señales 5 a 6, no son activadas a razón de que el reconocimiento por señales claves fue utilizado y reconoció WS_1 y WS_2 como posibles candidatos.
4. Se realiza el cálculo de los ranking por cada QoS de los WS_c, a partir de la Ecuación 4.8. La Tabla 4.4 muestra los resultados donde es especificado el factor de inserción en el ABB, seguidamente se encuentra el índice en inorden que determina el valor φ'
5. Se realiza el cálculo del ranking del WS a partir de los valores encontrados de cada φ', de tal forma que las Ecuaciones 4.10 y 4.11, reciben estos parámetros de entrada y establecen la categorización de los servicios candidatos. La Tabla 4.5 presenta los resultados de su evaluación. Acorde con estos valores, el WS_2 queda reconocido en el sistema como aquel

Tabla 4.2: Ejemplo de la especificación de RNF en un ACM

RNF	QoS_1	QoS_2	QoS_3	QoS_4	QoS_5	QoS_6	QoS_7	QoS_8	QoS_9
ACM	KS = 300	KS = 36	KS = 1	KS = 38	PS	PS	PS	PS	PS

Tabla 4.3: Ejemplo de módulos Γ en un proceso de reconocimiento

Γ	QoS_1	QoS_2	QoS_3	QoS_4	QoS_5	QoS_6	QoS_7	QoS_8	QoS_9
1	*true*	*true*	*true*	*true*	-	-	-	-	-
2	*true*	*true*	*true*	*true*	-	-	-	-	-

Tabla 4.4: Ejemplo del cálculo de rankings de QoS

WS_c	QoS_1		QoS_2		QoS_3		QoS_4	
i	IFMIN	φ'	IFMIN	φ'	IFMIN	φ'	IFMIN	φ'
1	-33.17	2	-1	2	-0.1	1	1	1
2	-39	1	-2	1	-0.1	1	1	1

que cumple a totalidad con los umbrales(Δ) dados en las restricciones de los RNF del modelo de composición.

Tabla 4.5: Ejemplo del cálculo de ranking de WS

WS_c	Rating	φ
1	-3.00	2
2	-4.00	1

6. Se obtienen las rutas de solución, para este caso en particular, el ACM sólo contaba con un servicio atómico y se establece una restricción para el servicio compuesto con $\Delta ACM = 80\,\%$. Realizando un cálculo trivial, el WS_1 tendrá una condición de calidad de $\Delta = 50\,\%$ y el WS_2 tendrá una condición de calidad de $\Delta = 100\,\%$. De esta forma, el servicio que satisface el ACM será el definido en WS_2.

4.8. MODELO ARQUITECTÓNICO DE AR_WSDS

Ar_WSDS propone un sistema basado en las estrategias de comunicación entre componentes de software, estableciendo a nivel de sistemas distribuidos, la interacción de un sistema base con el mecanismos de selección de servicios al lado del servidor.

A nivel arquitectónico Ar_WSDS, es definido dentro un patrón de comunicación *Service Discovery* donde se configura como una API para que otros componentes de software puedan descubrir servicios al lado del servidor[89]. El servicio de registro tiene la responsabilidad de cargar los diferentes repositorios y redireccionarlos a los servicios que irán a ser seleccionados. La Figura 4.15 presenta el patrón en el cuál opera Ar_WSDS.

Alrededor de ese problema se han desarrollado patrones de comunicación que utilizan terceros para orquestar la comunicación entre uno o varios componentes de software. Un sistema donde se utilicen brokers para comunicarse los diferentes componentes y necesite del proceso de selección de un servicio en particular.

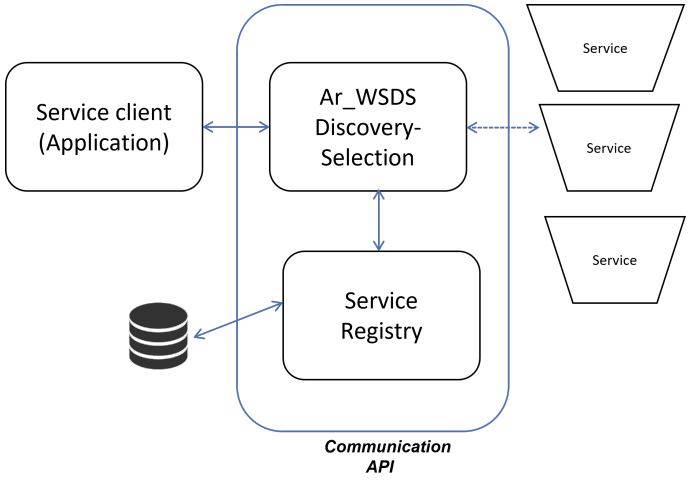


Figura 4.15: Patrón Service Discovery de Ar_WSDS.

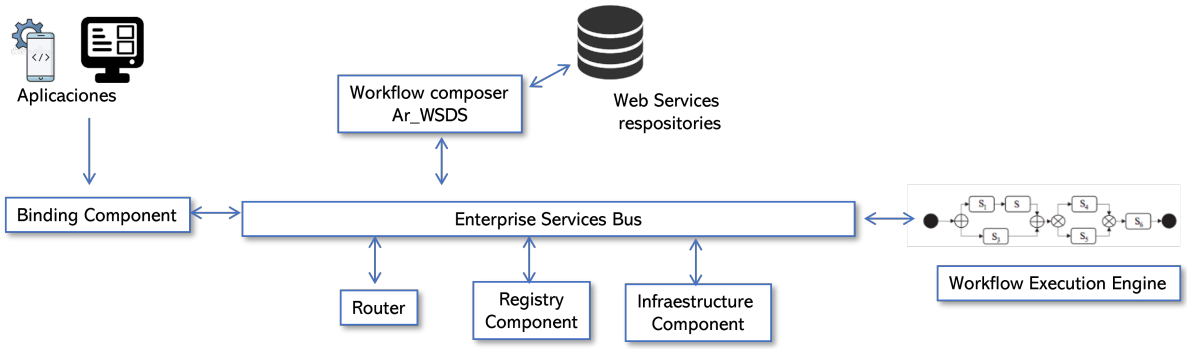


Figura 4.16: Arquitectura de un sistema de Composición usando Ar_WSDS.

La Figura 4.16 presenta la arquitectura general de un sistema basado en composición de servicios con la incorporación del componente Ar_WSDS. El sistema está conformado por:

El *binding component* que se encarga de serializar la composición en un workflow y sus especificaciones de RNFs basados en los criterios de calidad de servicio a nivel atómico y molecular.

El *Enterprise service bus* encargado de la gestión de servicios web, cuyo característica principal es la de servir como medio de integración y despliegue de los componentes software de forma distribuida. Toda comunicación con el sistema es realizada a través de él, esto asegura solo un punto de acceso para las aplicaciones y la fácil administración de los servicios que irán operar en el sistema. Constituye un middleware que define todas las políticas y reglas del negocio y conmuta las peticiones hacia los componentes propios de la aplicación que utilizará el servicio molecular o compuesto. A través de *Infrastructure component* se realiza la administración de recursos como datos, servicios en IaaS y la operación de diferentes canales de entrega que usarán los WS.

El *Workflow composer* selecciona los servicios atómicos que están en el modelo de compo-

sición. En este punto, el mecanismo Ar_WSDS descubre, clasifica y selecciona los servicios adecuados que satisfagan el ACM. Es posible que exista más de un WS que se adapte a la composición, Ar_WSDS proporcionará los servicios óptimos a partir del valor de calidad dado por el desarrollador.

El *Router* administra y garantiza los mensajes entre los servicios. Estos mensajes llevarán los requerimientos para el acceso a recursos propios que necesite cada servicio y que son operados por el ESB. En caso de fallas de un servicio, este componente informará a su entidad superior para seleccionar otra tupla de servicios(denominados SP en el entorno Ar_WSDS). El componente de registro proporciona interfaces para administración de los servicios según el tipo de composición realizada.

La ejecución es realizada por el motor de composición que evalúa el estado de cada uno de sus servicios a partir de su disponibilidad. Después de esto se realiza la instanciación de los servicios y la administración de los mensajes entre entidades pares(tarea realizada por el componente anteriormente descrito). Este componente también es el encargado de la actualización de métricas QoS que puedan afectar el acceso a un servicio y que deberán ser reevaluadas como restricciones entrantes para un mayor espacio de búsquedas en los repositorios de servicios.

4.8.1. Una mirada desde las arquitecturas de software

Como fue analizado en la sección 2.2, la arquitectura de diseño original de un WS es SOA donde los servicios constituyen un componente autónomo y con capacidades de calidad de servicio en diferentes proveedores; no obstante, desde la incorporación de cloud nuevos retos nacen en cuanto a los criterios de calidad que un WS tiene formalmente; esto es, por que cada proveedor depende de unos recursos que son predefinidos por los RNF de su cloud.

En contraste con SOA, cloud ofrece replicas de WS en diferentes instancias de sus proveedores, lo que ocasiona que los criterios QoS varíen en función de cada una de las plataformas donde el WS es desplegado. Estas instancias son casi exactas, por lo que, mecanismos como Ar_WSDS toman gran importancia a la hora de seleccionar servicios con similitud de QoS.

Ar_WSDS constituye un componente dentro del *workflow composer* que es definido dentro de una arquitectura de sistemas distribuidos, como una estrategia o patrón de comunicación a usar entre los componentes de software y ser embebido dentro un patrón que involucren terceros o proxys para la gestión de la selección.

En este mismo orden de ideas, cualquier arquitectura software permite diseñar una aplicación para su correcta estructuración y posterior despliegue, tomando en cuenta esto, el patrón de diseño de software desarrollado en Ar_WSDS ofrece un mecanismo transparente a ser operado sobre cualquier arquitectura; no obstante, es necesario aclarar que el sistema base debe poseer las características funcionales en las que opera la selección y la categorización usando el enfoque de reconocimiento de patrones de software.

4.8.2. Perspectiva general de la aplicación de Ar_WSDS en sistemas no monolíticos

Ar_WSDS opera bajo el paradigma cloud cuyo mecanismos se puede incorporar a cualquier solución software que requiera selección de componentes. El modelo da cabida a que un servicio pueda ser parte de un sistemas distribuido teniendo en cuenta las métricas que debe cumplir para este tipo de contextos.

El desarrollo de aplicaciones que utilizan este tipo de arquitecturas deben tener la capacidad de verificar la integridad y el funcionamiento de cada uno de los servicios, especialmente en sistemas que requieran procesos críticos como acceso a bancos y/o operaciones de autenticación. Por tal motivo, estos sistemas deben proveer información necesaria de las restricciones y preferencias de la utilización de cada uno de ellos.

Desde este punto de vista, Ar_WSDS a través de su descripción de composición puede ser operada desde la implementación de un sistema de composición a nivel de microservicios donde la evaluación de cada instancia de ellos se realiza a través de atributos de calidad, generalmente definidos por los criterios de rendimiento, disponibilidad y tiempo de respuesta[90].

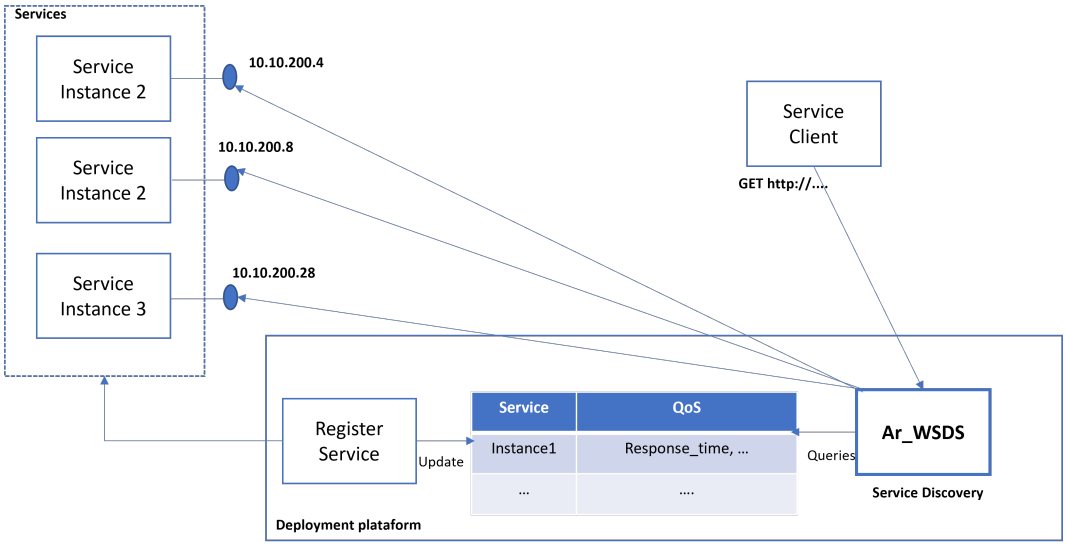


Figura 4.17: Ejemplo de Ar_WSDS bajo una arquitectura de microservicios
Adaptado de: [91]

La Figura 4.17 presenta un ejemplo de la incorporación de Ar_WSDS en un sistema basado en microservicios a través de la utilización del patrón *service discovery*. Acá se observa como el sistema es el responsable del registro de los servicios y de las condiciones QoS para cada una de sus instancias. Cada instancia tiene una localización de red y el mecanismo Ar_WSDS desplegado en el servidor selecciona el servicio adecuado para la petición del cliente. En este caso en particular, los módulos de reconocimiento son adaptados dinámicamente para que la respuesta al cliente sea la dirección IP de la instancia y está sea ejecutada a través de una petición GET.

La personalización de los QoS dependerá de las necesidades propias del cliente, donde él escogerá el tipo de reconocimiento a utilizar y los atributos de calidad que desea analizar para escoger su instancia.

4.9. CONCLUSIONES

Ar_WSDS alcanza sus objetivos de tres formas: el primero el reconocimiento sintáctico que consiste solo en aquellos WS cuya descripción de servicios corresponda sintácticamente a un servicio de algún repositorio definido en el conjunto WS_g; el segundo corresponde al reconocimiento clave, donde los servicios son reconocidos con umbrales definidos en intervalos cerrados para cada QoS; y finalmente, el tercero que corresponde al reconocimiento parcial, en este nivel el modelo de composición alcanzará a satisfacción todos los WS con los umbrales que alcancen el promedio de valores en sus QoS.

CAPÍTULO 5

Experimentación y Resultados

En esta sección se describirá la aplicación de los escenarios de experimentación sobre Ar_WSDS. Se especifica el dataset utilizado y el proceso de evaluación de cada estrategia de reconocimiento, adicional, se establece un marco conceptual de los criterios QoS con los que fueron desarrollados los experimentos.

De igual forma, se establece comparativa con dos enfoques de solución para analizar la eficacia del proceso de descubrimiento y selección de WS. Finalmente, se realiza la disertación de los hallazgos.

5.1. PREPARACIÓN DEL ENTORNO DE EXPERIMENTACIÓN

Ar_wsds fue probado con los WS registrados en el proyecto QWS Dataset [8] y un conjunto de datos de réplica para aumentar el espacio de búsqueda. El conjunto de datos describe nueve parámetros QoS: tiempo de respuesta, disponibilidad, rendimiento, fiabilidad, rendimiento, conformidad, mejores prácticas, latencia y documentación. Estos fueron analizados y colocados según su naturaleza en dos grupos que correspondían a valores que deben ser maximizados o minimizados a partir de lo presentado en [86].

Ar_wsds toma como escenarios de prueba un conjunto de WS en workflow secuenciales para analizar la efectividad y rendimiento de la solución propuesta. De la Figura 4.14 se presenta un modelo genérico de composición con cuatro WS y estos son reconocidos usando Ar_wsds con reconocimiento KS y un (ΔACM). Esta forma genérica de ver los resultados ofrece el panorama de la evaluación a nivel de las entradas del sistema y sus salidas. La matriz MC, presenta las SP ordenadas según su criterio de calidad a partir del ΔACM que se ha registrado, en consecuencia de esto, la matriz sólo contendrá aquellos SP que cumplan con está condición. No obstante, para efectos de simulación del sistema y con el objetivo de servir como mediador digital, en el desarrollo del sistema se ha almacenado temporalmente todas las SP. Está condición no es tenida en cuenta para la comparación entre enfoques.

Los experimentos son ejecutados con una aplicación realizada en Python y carga dinámica del dataset correspondiente a [8] con la adición de los datos de réplicas conforme a la naturaleza

del dataset original. Esto permite operar múltiples cardinalidades para el conjunto WS_g. Los experimentos son diseñados en los siguientes escenarios:

Escenario 1. Reconocimiento por señales claves, su fin es realizar evaluación de la estrategia KS.
Escenario 2. Reconocimiento por señales claves, su fin es realizar evaluación de la estrategia KS.
Escenario 3. Reconocimiento por señales claves y parciales, su fin es realizar evaluación de la estrategia KS versus PS.
Escenario 4. Reconocimiento clave y parcial sobre un caso de estudio, su fin es realizar la evaluación de Ar_WSDS en una composición con requisitos funcionales y no funcionales en un entorno real de trabajo.
Escenario 5. Ar_WSDS frente a dos enfoques de selección, su fin es analizar la efectividad del sistema propuesto frente a dos estrategias computacionales.
Escenario 6. Evaluación de WS por sus criterios QoS, su objetivo es el de analizar como afectan los criterios QoS de un WS para el proceso de selección. Se realiza la evaluación por cada QoS del WS_T.

El proceso de evaluación utiliza las estrategias por reconocimiento KS y PS en cada escenario, cada uno de los cuales considera un número variable de QoS activos (señales) y de $WS_g \in ACM$. Las rutas de solución dependen de la cantidad de WS_c del conjunto de entrada, matriz definida en la Ecuación 4.6.

El despliegue de la aplicación que permite realizar los experimentos tiene una arquitectura hardware virtualizado en DigitalOCean con 4GB de RAM, 1 CPU, sistema operativo Linux Ubuntu 18.04.3 LTS y Python versión 3.6. Son desarrollados dos frontends para la simulación del sistema, el primero en modo consola para la carga masiva de servicios; el segundo, una vista web con la funcionalidad de seleccionar a través de componentes gráficos los WS y cada uno de sus patrones. Los resultados son presentados en un archivo de texto ASCII o sobre una tabla dinámica que puede ser fácilmente exportable a una hoja de cálculo o a un formato pdf.

5.2. PARÁMETROS EXPERIMENTALES

Son seleccionados 100 servicios con réplicas para forzar al algoritmo a realizar búsquedas y selecciones exhaustivas. La tasa de réplica de cada WS_i varía entre 20 y 50 sobre diferentes proveedores. En consecuencia, el conjunto WS_g corresponde a los datos del repositorio QWS Dataset y a las replicas generadas.

La Tabla 5.1, describe los criterios QoS y sus características, con está información los módulos de reconocimiento(Γ) definen sus valores de análisis basados en la naturaleza del QoS en intervalos que pueden ser máximos o mínimos. Cada parámetro QoS es identificado canónicamente desde P_1 a P_9. Por ejemplo, P_1 su valor de calidad es analizado usando mínimos (asignado el carácter ”-”) y corresponde a la característica QoS de *tiempo de respuesta*.

Los experimentos son realizados en cinco escenarios, anteriormente mencionados, cada cual utiliza reconocimiento a través de valores constantes de ΔACM, y para cada servicio de ΔKS y

ΔPS. El primer y segundo escenario, evalúan el reconocimiento por señales claves y parciales para establecer una métrica de comparación entre el tiempo de procesamiento y la cantidad de SP que cada estrategia logra reconocer.

El tercer escenario combinaba el reconocimiento clave y el parcial y presenta su tasa de éxito. El cuarto escenario evalúa el reconocimiento con un caso de estudio de un entorno de composición en tiempo real, valores tomados directamente del repositorio y finalmente, el último escenario realiza la comparación entre dos enfoques en el proceso de clasificación y ranking.

Tabla 5.1: Descripción de parámetros QoS para selección de WS

Parámetro	**Descripción**	**Unidad**	**Max(+)/Min (-)**
Tiempo de Respuesta (P_1)	Tiempo que tarda en enviar una solicitud y recibir la respuesta.	ms	−
Disponibilidad (P_2)	Porcentaje de tiempo en que un WS está disponible para recibir invocaciones	%	+
Rendimiento (P_3)	Cantidad de invocaciones(I) de un WS sobre su unidad de tiempo(seg) .	I/seg	+
Tasa de éxito (P_4)	Porcentaje de éxito tras la invocación de un servicio.	%	+
Fiabilidad (P_5)	Porcentaje de fallos en un periodo de tiempo	%	−
Cumplimento (P_6)	Porcentaje de cumplimiento del WSDL basado en la especificación formal de la W3C.	%	+
Latencia (P_7)	Tasa de demora en procesar una invocación.	%	−
Mejores prácticas (P_8)	Tasa de cumplimiento de la especificación del servicio bajo el estándar WS-I.	%	+
Documentación (P_9)	Tasa de cumplimiento de la documentación proporcionada del servicio en su proveedor sobre el WSDL.	%	+

Adaptado de: [8]

5.2.1. Análisis de parámetros QoS

Las métricas utilizadas en calidad de servicio son descritas en [8], en él es definido nueve parámetros, los cuales sirvieron de base para la prueba del sistema. Para la incorporación de estos valores en los escenarios de prueba, se toma en cuenta la naturaleza de la métrica (máximo o mínimo) con valores discretos mayores que 0.

Para la eliminación del ruido en la información que puede ocurrir en la etapa de *Obtención y normalización* de datos mostrada en la Figura 4.2 y cada parámetro QoS es validado dentro de su módulo de reconocimiento (Γ). De esta forma, se establece un valor $QoS_{gi,j}$ como se muestra en la Figura 5.1, donde:

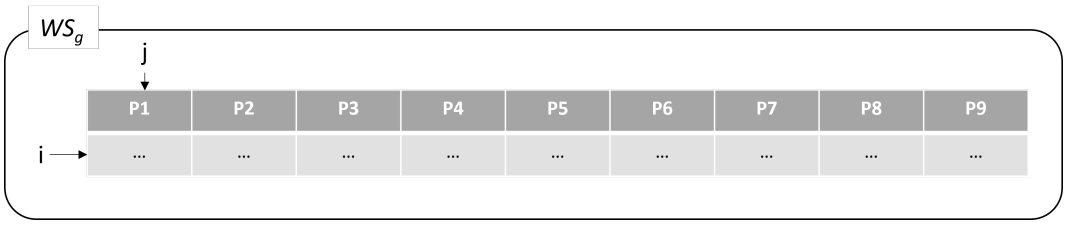


Figura 5.1: Especificación de QoS para validación de un servicio candidato.

El valor de cada QoS es calculado a partir de su media aritmética, definida en la Ecuación 5.1.

- QoS_g es un conjunto de métricas QoS para un servicio WS de entrada.
- i es el índice WS_g
- j es el valor QoS para el WS_g. Todos los valores con índice j pertenecen a una misma característica QoS.

$$Metric(QoS_{i,j}) = \frac{\left(\sum_{j=1}^{9}\left(QoS_j\right)\right)}{|QoS_i|} \tag{5.1}$$

Tomando en cuenta esté valor de métrica, cada criterio QoS es evaluado si es mínimo o máximo.

$$isValid\_Min(QoS_{i,j}) = \left(Metric(QoS_{i,j}) > 0 \wedge Metric(QoS_{i,j}) <= QoS_i\right) \tag{5.2}$$

$$isValid\_Max(QoS_{i,j}) = \left(Metric(QoS_{i,j}) > 0 \wedge Metric(QoS_{i,j}) >= QoS_i\right) \tag{5.3}$$

La Tabla 5.2 presenta la especificación de la función de validación que es utilizada para cada criterio QoS con las ecuaciones 5.2 y 5.3. La utilización de esté proceso permite filtrar el ruido que pueda existir en los datos de entrada; así como, únicamente tomar en cuenta los valores QoS que estén en un rango válido y que puedan operar con los módulos de reconocimiento Γ.

Tabla 5.2: Especificación de la función de validación de métricas QoS para selección de WS

Parámetro	isValid_Min	isValid_Max
Tiempo de Respuesta (P_1)	✓	
Disponibilidad (P_2)		✓
Rendimiento (P_3)		✓
Tasa de éxito (P_4)		✓
Fiabilidad (P_5)	✓	
Cumplimento (P_6)		✓
Latencia (P_7)	✓	
Mejores prácticas (P_8)		✓
Documentación (P_9)		✓

5.2.2. Especificación de patrones y señales

En el primer y segundo escenario se utilizaron tres conjuntos de datos, con cardinalidades de 10, 50 y 100 WS con dos, cuatro y nueve señales, respectivamente. Estos fueron agrupados de la siguiente manera: dos señales utilizaron P_1 y P_2 , cuatro señales utilizaron de P_1 a P_4 , y en nueve señales se utilizaron P_1 a P_9 QoS.

El tercer escenario utiliza 50 WS, y cada prueba combina los dos tipos de reconocimiento. Se definen un conjunto de pruebas para combinar P_1 a P_9 QoS. El sistema comenzó su reconocimiento utilizando relaciones de 2 a 7, 4 a 5 y 1 a 8 señales parciales y claves, respectivamente. Otras pruebas invirtieron estas relaciones para que el sistema reconociera utilizando primero las señales clave. El cuarto escenario utiliza tres WS de un ACM.

Por último, el quinto escenario evaluá Ar_WSDS con respecto a otros enfoques. Las pruebas se realizaron con un conjunto de nueve servicios seleccionados, todos ellos con réplicas. La prueba se realizó utilizando las siguientes condiciones iniciales para Ar_WSDS:

- La fiabilidad, la disponibilidad y el tiempo de respuesta fueron las únicas señales activadas;
- Se estableció un $\Delta ACM = 90\,\%$ para elegir los WS del top del ranking; y
- El reconocimiento por señales clave utilizó valores de WS muy cercanos a sus valores límite, esto permite al sistema seleccionar los WS mejor calidad.

5.3. ESCENARIO DE RECONOCIMIENTO POR SEÑALES CLAVES (KS)

El reconocimiento KS define la búsqueda y la selección de forma restrictiva, esto es, por que sus intervalos de reconocimiento son cerrados; por lo tanto, cuanto mayor sea el número de señales activas, menor será la cardinalidad de la búsqueda. Los módulos de reconocimiento KS (Γ) son creados dinámicamente y las señales activan su funcionamiento, según el valor definido por en sus respectivos ΔKS. Estos valores se eligieron a partir de una configuración que representará el mayor de número WS reconocidos.

Tabla 5.3: Resultados del tiempo de ejecución usando KS

WS_T	**KS**	WS_C	**Tiempo de Ejecución(seg)**	**SP**
10	2	17	0.5	8
50	2	129	5.82	34
100	2	1047	122.3	166
10	4	10	0.3	1
50	4	81	3.21	21
100	4	421	39	58
10	9	10	0.17	1
50	9	50	2.34	1
100	9	100	6.02	1

La Tabla 5.3, presenta los resultados de la evaluación de las señales clave. La primera columna representa los servicios de la composición (Ecuación 4.1). La segunda columna el número de señales, la tercera columna el tiempo de ejecución medido en segundos para salidas por el frontends en modo consola, este tiempo es tomado después de la carga de WS_g; y la columna SP corresponde al número de n-tuplas de solución bajo $\Delta ACM \geq 80$.

5.4. ESCENARIO DE RECONOCIMIENTO POR SEÑALES PARCIALES (PS)

Se utiliza la misma definición de escenarios del reconocimiento KS, tomando en cuenta que este tipo de reconocimiento tendrá una mayor cantidad de WS_c y por consiguiente las SP aumentarán. La función $\beta(\varphi)$ depende del valor promedio por cada $QoS_{i,j}$ a reconocer, de esta forma, el procesamiento en Ar_WSDS en esté tipo de reconocimiento aumenta considerablemente.

Tabla 5.4: Resultados del tiempo de ejecución usando PS

WS_T	**PS**	WS_C	**Tiempo de Ejecución(seg)**	**SP**
10	2	180	2.13	42
50	2	418	3.25	56
100	2	5620	427.2	402
10	4	83	3.07	28
50	4	137	4.23	39
100	4	1827	201.3	193
10	9	32	1.2	13
50	9	63	4.22	19
100	9	731	71.2	67

La Tabla 5.4 presenta los resultados de la fase de reconocimiento por señales parciales. En

ella se muestra cómo la cantidad de servicios candidatos aumenta. Esta condición subyace del hecho que algunos WS pueden tener valores altos o iguales en sus QoS generando mayor número de WS que pertenecen a ese intervalo de su valor promedio, resultado diferente a la limitante impuesta por el reconocimiento KS. Por otra parte, los resultados muestran una tendencia a encontrar menor número de rutas de solución cuando las 9 señales están activas para su reconocimiento. Adicionalmente, para cardinalidades muy altos del conjunto WS_c, el tiempo de procesamiento para encontrar las SP tiende en aumentar, el $\beta\,(\varphi)$ manejado fue de 1 y 2 , para dar mayor cobertura al espacio de solución.

5.5. ESCENARIO DE RECONOCIMIENTO POR SEÑALES CLAVES Y PARCIALES

El sistema Ar_WSDS define su modelo de reconocimiento basado en el descubrimiento de patrones definido Figura 4.8 y Ecuación 4.6. En ellos se establece si un patrón no puede ser reconocido por señales claves este será reconocido por señales parciales. Se realizaron pruebas del porcentaje de éxito de cada uno de los reconocimientos usando 9 parámetros QoS, los resultados son presentados en la Tabla 5.5.

Para cada experimento, el número de señales es de 9 , con variaciones entre las señales parciales y las claves, de manera que el sistema podía reconocer un grupo de señales por KS o bien por PS. Todos los módulos de reconocimiento se crearon dinámicamente en este escenario. Esta configuración significa que, a medida que aumentan las restricciones en ambos reconocimientos, la tasa de éxito del reconocimiento disminuye. En la práctica, esta condición implica que, si se solicita un servicio con altas restricciones en sus requisitos no funcionales, pocos servicios cumplirían con altos niveles con todos los QoS configurados.

Tabla 5.5: Tasa de éxito usando reconocimeinto KS y PS

Test	PS	KS	Tasa de éxito PS	Tasa de éxito KS
1	2	7	82	18
2	4	5	67	33
3	1	8	91	9
4	7	2	21	79
5	5	4	41	59
6	8	1	11	89

5.6. ESCENARIO CON UN CASO DE ESTUDIO

La composición servicios define aplicaciones que a través de la combinación de varios servicios puede dar solución a un requerimiento funcional complejo. Para el caso de estudio, el requerimiento funcional complejo consiste en el almacenamiento de mensajería virtual sobre su servidor de comunicaciones, para esto el desarrollador realiza la siguiente especificación de requisitos:

- Son requeridos tres WS con identificación: DOTSGeoPhone, BINDService y WebStoreService.
- La composición de servicios debe asegurar un nivel de calidad del 90 %.
- Los RNFs en términos QoS más representativos serán: el tiempo de respuesta, disponibilidad , rendimiento y tasa de éxito. Los demás criterios son opcionales.

El workflow de la composición es mostrado en la Figura 5.2 y cada WS será reconocido por cuatro señales claves y cinco parciales a través de las métricas definidas en la Tabla 5.6 .

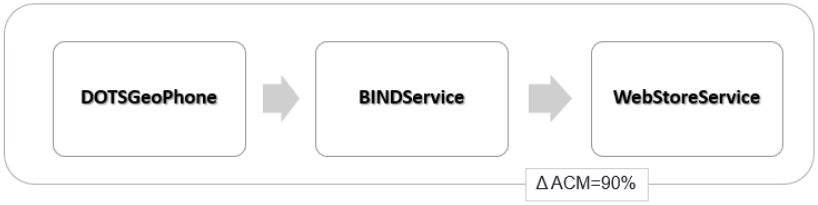


Figura 5.2: Workflow de la composición para el caso de estudio.

Tabla 5.6: Descripción de los WS del ACM para el caso de estudio

Nombre del Servicio	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
BINDService	KS = 300	KS = 36	KS = 0.8	KS = 36	PS	PS	PS	PS	PS
DOTSGeoPhone	KS = 106	KS = 90	KS = 187	KS = 100	PS	PS	PS	PS	PS
WebStoreService	KS = 300	KS = 60	KS = 60	KS = 60	PS	PS	PS	PS	PS

Basados en la invocación de la función de reconocimiento M(), la Tabla 5.7 presenta el resultado de los WS obtenidos del conjunto WS_g.

Tabla 5.7: Descripción de los WS del caso de estudio

Nombre del Servicio	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
BINDService	266.83	36	0.9	37	60	89	69	15.91	7
	261	36	0.9	37	60	89	69	13.67	12
DOTSGeoPhone	102	90	18.6	97	73	78	80	1	92
	107.4	85	19	95	73	78	80	0.8	92
WebStoreService	251.07	56	52	58	73	67	80	133.43	31

El método de reconocimiento de Ar_WSDS se describe a continuación:

1. Creación de la matriz QoS, su función es la de establecer los valores que van procesar los módulos de reconocimiento (Γ).

Global ranking	Response Time	Response Time	Throughput	Successability	Reliability	Compliance	Best Practices	Latency
BINDService - Input: Keys Recognition(4) - Partial recognition(0) Output: Web Services recognition by: key signal								
1	Rank: 1 261	Rank: 1 36	Rank: 1 0.9	Rank: 1 37	Rank: 0 60	Rank: 0 89	Rank: 0 60	Rank: 0 13.37
2	Rank: 2 266.83	Rank: 1 36	Rank: 1 0.9	Rank: 1 37	Rank: 0 60	Rank: 0 89	Rank: 0 69	Rank: 0 15.91

Figura 5.3: Resultado del reconocimiento del servicio BINDService

$$QoS = \begin{bmatrix} 266{,}83 & 36 & 0{,}9 & 37 & 60 & 89 & 69 & 15{,}91 & 7 \\ 261 & 36 & 0{,}9 & 37 & 60 & 89 & 69 & 13{,}67 & 12 \\ 102 & 90 & 18{,}6 & 97 & 73 & 78 & 80 & 1 & 92 \\ 107{,}4 & 85 & 19 & 95 & 73 & 78 & 80 & 0{,}8 & 92 \\ 251{,}07 & 56 & 52 & 58 & 73 & 67 & 80 & 133{,}43 & 31 \end{bmatrix}$$

2. Reconocimiento para BINDServices, el resultado de le ejecución del sistema es presentado en la Figura 5.3, fueron reconocidos dos candidatos. Sus cuatro señales KS activaron los módulos Γ con su clasificación respectiva. Sin embargo, el servicio con el mejor ranking(φ) se lista de primero. Este proceso da como resultado una señal del servicio 2 (tiempo de respuesta=$QoS_{1,0} = 261$), que tenía un valor de tiempo de respuesta más apropiado para los criterios de calidad solicitados.
3. Reconocimiento para DOTSGeoPhone, fue reconocido un sólo servicio , presentado en la Figura 5.4, son pre-clasificados por M() dos servicios, por lo que permite al sistema elegir fácil y rápidamente. Las señales clave estaban fuera del valor ΔKS, y el sistema realizó el reconocimiento utilizando señales parciales, estas activan sus módulos, seleccionando el servicio que cumpla con cada ΔPS; en este caso, se descartó el primer servicio DOTSGeoPhone, ya que su valor de $QoS_{2,8} = 1$ estaba fuera del rango medio.

Global ranking	Response Time	Availability	Throughput	Successability	Reliability	Compliance	Best Practices	Latency	Documentation
DOTSGeoPhone - Input: Keys recognition (5) Output: Web services recognition by: Partial signal									
1	Rank: 0 107.4	Rank: 0 85	Rank: 0 19	Rank: 0 95	Rank: 0 73	Rank: 0 78	Rank: 0 80	Rank: 0 0.8	Rank: 0 92

Figura 5.4: Resultado del reconocimiento del servicio DOTSGeoPhone.

4. Reconocimiento para WebStoreService, el servicio fue reconocido como único por la función M(). El reconocimiento clave no es activado ($QoS_{4,3}$ fuera del valor ΔKS.), sus

señales fueron analizadas por reconocimiento parcial y es seleccionado el servicio, su resultado es presentado en la Figura 5.5.

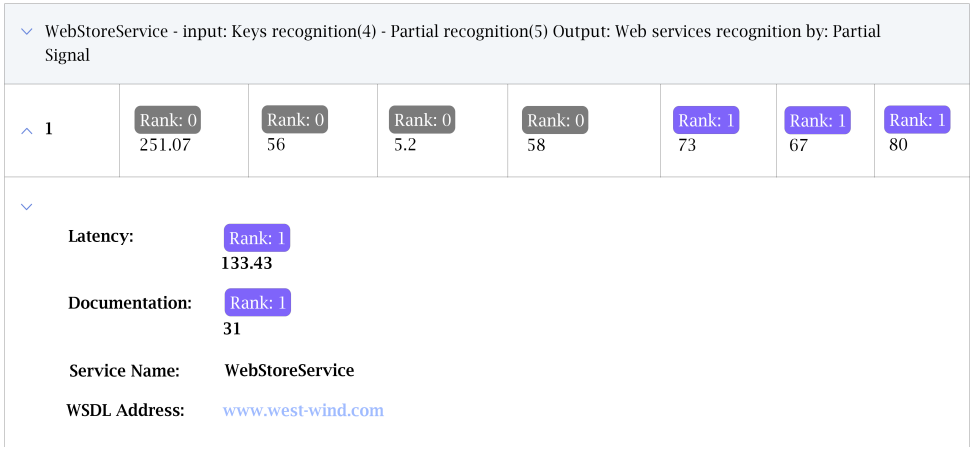


Figura 5.5: Resultado del reconocimiento del servicio WebStoreService.

5. Se genera la matriz $MC[][]$ y se procesaron los SP, para las tuplas de servicios que satisfacen $\Delta WSC = 90\,\%$. La Figura 5.6 muestra el SP correspondiente al requisito solicitado, de cada servicio es mostrado la información relativa a su proveedor, su especificación WSDL y su ranking. Esto permite al desarrollador analizar diferentes opciones para elegir los servicios según la configuración de criterios de calidad requeridos.

Figura 5.6: Rutas de solución del caso de estudio.

5.7. ESCENARIO DE AR_WSDS CON OTROS ENFOQUES DE SELECCIÓN Y RANKING DE SERVICIOS

Ar_WSDS se comparó con OAEQoS [2] y con un algoritmo genérico de descubrimiento y selección (por sus siglas en inglés GDSA). El primer algoritmo ofrece un método sencillo para calcular y analizar parámetros QoS, su estrategia incluye un factor de comprobación de acceso y rendimiento del servicio; en el segundo, se obtenía una solución trivial mediante búsquedas exhaustiva. El proceso se lleva a cabo para analizar cada servicio y la forma de elegir sus candidatos, frente los servicios esperados en la selección. OAEQoS se configura para que sus parámetros de entrada y salida generarán la lista de servicios web candidatos.

Los algoritmos 5.1 y 5.2 presentan las estrategias OAEQoS y GDSA, respectivamente. Cada algoritmo utilizó entradas similares a las de Ar_WSDS. Sin embargo, el ACM se descompuso

Algoritmo 5.1: Adaptación del algoritmo OAEQoS

```
Input: service_list,service_list_required, and QoS_list_required
Result: service_list_candidate with Qos and rank
Read service_list,service_list_required, and QoS_list_required;
Select WS from service_list, considering service_list_required;
for s ∈ service_list do
    if s invoked process = true then
        Evaluate invoked failed and bounced;
        Update reliability, availability, and response_time;
        Calculate OAEQoS value;
        if OAEQoS is within the expected QoS_list then
            Calculate rank with the value of OAEQoS;
            Append s and its rank to service_list_candidate;
        end
    else
        s is rejected as a candidate;
    end
end
```

en datos más específicos, para cargar la lista de WS requerida y cada criterio QoS. Se utilizó el conjunto de datos QWS de entrada (*service_list*) y fueron utilizados los parámetros de fiabilidad, disponibilidad y tiempo de respuesta.

Algoritmo 5.2: Algoritmo genérico de búsqueda y selección de servicios web.

```
Input: service_list,service_list_required, and QoS_list_required
Result: service_list_candidate with Qos and rank
Read service_list,service_list_required, and QoS_list_required;
Select WS from service_list, considering service_list_required;
for s ∈ service_list do
    Calculate average reliability, availability, and response_time for s;
    Calculate expected average from QoS_list_required;
    if average is within the expected average then
        Append s and its average to service_list_candidate;
    else
        s is rejected as a candidate;
    end
end
Sort service_list_candidate for each candidate service by its average value;
Rank based on the candidate's position in service_list_candidate;
```

Se realizan dos pruebas, la primera usa reconocimiento por señales claves y la segunda por

parciales. La Tabla 5.8 muestra los nombres de los servicios web que hacen parte del ACM,un identificador y la cantidad de servicios, esté último detalla la cantidad de replicas, el objetivo es el de aumentar el espacio de búsqueda de los tres algoritmos. El conjunto tiene una cardinalidad de $|WS_g| = 137$ servicios.

Tabla 5.8: Descripción de WS usados para la prueba de enfoques

ID WS	Nombre del Servicio	Cantidad
1	GoogleSearchService	43
2	MathService	9
3	AWSECommerceService	25
4	OnlineMessenger	5
5	DownloadService	17
6	LandmarkService	8
7	WSDLInteropTestDocLitService	13
8	AmazonSearchService	13
9	TimeService	4

Las Tablas 5.9 y 5.10 muestran la evaluación del proceso de selección en relación con los servicios esperados. Los datos de la Tabla 5.9, sólo incluyen los servicios reconocidos y determina los que cumplen con las restricciones de calidad registradas. Este resultado no proporciona un análisis del orden, según el ranking del servicio; sólo analiza la cantidad y la relevancia del servicio con los WSs esperados.

Tabla 5.9: Resultados del proceso de selección

ID WS	Ar_WSDS KS	Ar_WSDS PS	OAEQoS	GDSA	WS esperados
1	4	5	2	20	4
2	1	1	1	5	1
3	6	6	4	19	6
4	1	1	2	3	1
5	3	4	2	8	4
6	2	2	1	5	2
7	5	5	5	8	5
8	3	3	1	8	3
9	1	1	2	2	1

Tabla 5.10: Cantidad de errores en el proceso de selección de servicios

ID WS	Ar_WSDS KS	Ar_WSDS PS	OAEQoS	GDSA
1	0	1	2	16
2	0	0	0	4
3	0	0	2	13
4	0	0	1	2
5	1	0	2	4
6	0	0	1	3
7	0	0	0	3
8	0	0	2	5
9	0	0	1	1

En relación a estos resultados, se analiza finalmente, el orden de los servicios según las rutas o tuplas de solución. El orden corresponde a su ranking, que determina la posición en en el proceso de clasificación los candidatos seleccionados; por ejemplo, WS_1 corresponde al servicio GoogleSearchService, cuyos candidatos están en las posiciones 3, 15, 28 y 31. Los resultados se presentan en forma de dos tuplas *(WS ID, ranking)*, con clasificaciones que van de 1 *hasta n*; si el valor del ranking es 0, indica que el servicio no fue reconocido entre los candidatos por Ar_WSDS.

Tabla 5.11: Resultados de ranking de WS

ID WS	Ar_WSDS KS	Ar_WSDS PS	OAEQoS	GDSA
1	(3,1)-(15,2)-(28,2)-(31,2)	(3,1)-(15,2)-(28,2)-(31,2)	(3,1)-(15,2)-(28,3)-(31,3)	(3,1)-(15,12)-(28,6)-(31,4)
2	(5,1)	(5,1)	(5,1)	(5,3)
3	(19,1)-(2,1)-(8,1)-(20,2)-(10,2)-(9,2)	(19,1)-(2,1)-(8,1)-(20,2)-(10,2)-(9,2)	(19,2)-(2,3)-(8,0)-(20,2)-(10,2)-(9,0) *	(19,1)-(2,2)-(8,3)-(20,2)-(10,1)-(9,2)
4	(2,1)	(2,1)	(2,1)	(2,1)
5	(8,1)-(2,1)-(11,2)-(4,2)	(8,1)-(2,1)-(11,2)-(4,3)	(8,1)-(2,0)-(11,2)-(4,0) *	(8,1)-(2,1)-(11,2)-(4,2)
6	(1,1)-(5,2)	(1,1)-(5,2)	(1,1)-(5,0) *	(1,1)-(5,2)
7	(2,1)-(3,1)-(11,2)-(7,3)-(6,3)	(2,1)-(3,1)-(11,2)-(7,3)-(6,2)	(2,1)-(3,1)-(11,2)-(7,3)-(6,2)	(2,1)-(3,2)-(11,3)-(7,3)-(6,2)
8	(10,1)-(4,1)-(11,2)	(10,1)-(4,1)-(11,1)	(10,0)-(4,1)-(11,0) *	(10,1)-(4,2)-(11,2)
9	(3,1)	(3,1)	(3,1)	(3,1)

$*$ Error en el reconocimiento de WS_c

5.8. ESCENARIO PARA LA VALIDACIÓN DE CALIDAD DE SERVICIO DE UN WS

Dentro del proceso de selección desarrollado por Ar_WSDS se encuentra el mecanismo de ranking probado en el escenario anterior, adicional a esto, se realizaron pruebas usando la estrategia descrita en la sección 4.6. Estás pruebas ofrecieron un panorama general del comportamiento que tienen los QoS para los WS.

Se realiza pruebas utilizando el servicio *GoogleSearchService* que es aquel que contiene la mayor cantidad de replicas en el repositorio. El simulador[1] del sistema puede operar con cualquier servicio que se encuentre descrito en QWS, donde se genera un ranking para cada criterio QoS.

Tabla 5.12: Categorización de Qos para el WS GoogleSearchService

Ranking	**Cantidad de WS por Criterio**								
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
1	1	9	3	10	3	2	2	2	5
2	1	1	1	7	-	-	-	-	10
3	1	6	1	7	-	-	-	-	6
4	1	1	1	8	-	-	-	-	5
5	1	1	1	2	-	-	-	-	1

Los resultados de la aplicación del ranking se pueden observar en los anexos,Figuras 7.1,7.2,7.3,7.4, 7.5, 7.7,7.8 y 7.9. Cada parámetro QoS es analizado y colocado en un ranking general. Son listados los 5 primeros ranking que corresponden aquellos WS que están por encima de un $\Delta = 90\,\%$. La Tabla 5.12 presenta los resultados del ranking por criterio , está representación ofrece un panorama de la calidad de los servicios expuestos en Ar_WSDS, donde se puede destacar:

- Para el parámetro QoS de Tiempo de Respuesta (P_1), existe un sólo WS por ranking que cumple con la condición de la métrica. En esté caso, sólo cinco servicios están con valores deseables en su tiempo de respuesta.
- Para el parámetro QoS de Disponibilidad (P_2) , existe 9 WS que tienen el mismo valor de métrica y estos se encuentran en el ranking 1. El ranking 2, 4 y 5 con un solo WS , mientras que el ranking 3 está compartido por 6 servicios, con un total de 18 WS que cumplen con la condición de calidad.
- Para el parámetro QoS de Rendimiento (P_3) existen 7 WS en el primer puesto, 7 en el segundo y tercero, 8 en el cuarto y dos en el quinto. Está condición es importante resaltar que existirán alrededor de 34 WS que cumplen con el criterio de calidad.
- Para los parámetro QoS de Tasa de éxito (P_4), Fiabilidad (P_5), Cumplimento (P_6) y Latencia (P_7). Sólo existen WS de ranking 1. Está condición se da cuando los demás servicios están por debajo de la condición de validación evaluada por las Ecuaciones 5.2 y 5.3.

[1]En la siguiente URL se puede encontrar el simulador online: http://arwsds.madarme.co/search-service?service_name=GoogleSearchService

- Para el parámetro QoS de Documentación (P_9) existen 5 WS que comparten el ranking 1, 10 el ranking 2 , 6 el ranking 3 , 5 el ranking 4 y 1 el ranking 5. Se puede observar que la diferencia en categorización hace que existan más servicios en el segundo puesto.

Los resultados de todas los escenarios son discutidos y analizados en la siguiente sección.

5.9. DISCUSIÓN

El descubrimiento y selección de WS en entornos cloud ha generado gran interés en el área de tratamiento QoS ya que permite ofrecer a los usuarios servicios de alto rendimiento y un mejor acceso a los proveedores. El enfoque propuesto da una perspectiva del comportamiento de los servicios y el cambio dinámico que ocurre en sus parámetros de calidad, que difieren significativamente de proveedor a proveedor en los cuales los servicio son desplegados. La redundancia introducida en el conjunto de datos mostró las diferencias entre los servicios descubiertos y los seleccionados. Así, el tiempo de ejecución de la estrategia Ar_WSDS aumentó en función del número de WS replicados. Sin embargo, en un entorno natural, estos escenarios pueden identificar servicios con información actualizada que presentan marcadas diferencias entre sus valores de QoS, donde el reconocimiento proporcionado por Ar_WSDS puede dar lugar a considerables mejoras en su tiempo de ejecución.

Los ACM son considerados como un problema en un workflow secuencial; sin embargo, hay varios tipos de flujos: selección, paralelo y cíclicos, estos dependen principalmente de un QoS global (parámetros de QoS que el servicio compuesto debe cumplir), estos a través de técnicas se pueden convertir en workflows secuenciales, procesos que pueden generar más de un ACM derivado de esté tipo de composiciones. La mayoría de las soluciones reportadas en la literatura utilizan QoS global [2, 4, 66, 68, 69, 72], y cómo los servicios contribuyen a alcanzar el objetivo, a diferencia de ellos, Ar_WSDS evalúa cada servicio en un nivel de personalización atómico .

El QoS global en nuestro enfoque, se toma como un valor de calidad que debe ser alcanzado por cada camino de solución (conjunto de servicios candidatos para cada nodo de la composición-"SP") creado por nuestro sistema. Este tipo de adaptación permite mejorar el seguimiento de los RNF de los servicios individuales. Esta característica se denota en las composiciones dinámicas que requieren cambios estructurales en la funcionalidad de un servicio compuesto sin alterar el RF para el que fue creado.

La Figura 5.7 presenta la comparación por reconocimiento por señales claves y parciales. En ella se toma el promedio de las rutas de solución encontradas para 2, 4 y 9 señales activas respectivamente. En consecuencia, si la cantidad de señales aumenta la cantidad de rutas de solución disminuirá, siendo menor el valor de rutas descubiertas por señales claves ya que este tipo de reconocimiento es más restrictivo y depende del proceso ejecutado en el módulo $\Gamma_{i,j}$ según la naturaleza del QoS.

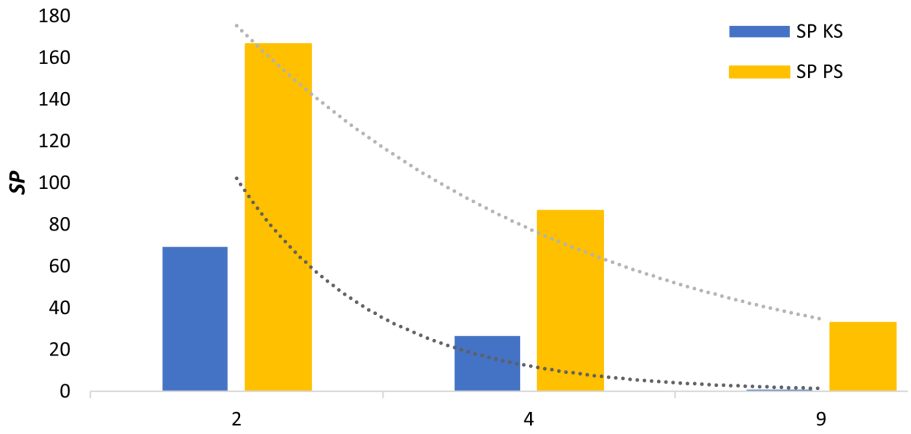


Figura 5.7: Comparación entre reconocimiento KS y PS.
(Escenario 1 y 2)

A medida que los parámetros QoS aumenten para la selección, el espacio de búsqueda se irá reduciendo, luego encontrar rutas de solución para el modelo de composición de entrada se realizará en tiempos de ejecución más eficientes, en contraste de soluciones con búsquedas secuenciales con menores restricciones QoS, ya que existirán mayor número de WS_c.

El tiempo de reconocimiento KS y PS tienen valores símiles de ejecución, como se puede notar en las Figuras 5.8 y 5.9. La razón del aumento de procesamiento va ligada con la cardinalidad del conjunto WS_T, así mismo, como se ha venido exponiendo, el aumento en la cantidad de QoS redunda en la disminución de servicios candidatos , principalmente en el reconocimiento KS, en contraste con el reconocimiento PS que siempre tendrá un mayor conjunto de servicios candidatos.

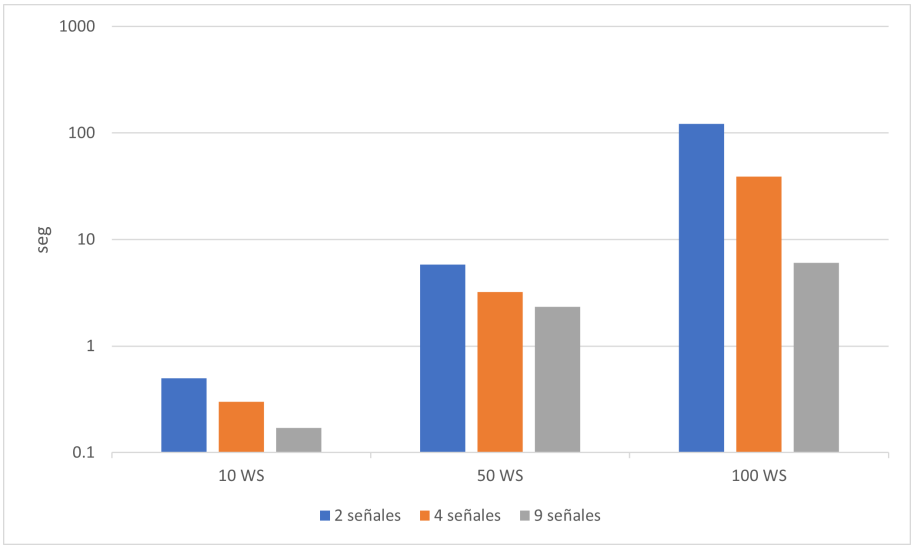


Figura 5.8: Relación del tiempo en reconocimiento KS.
(Escenario 1)

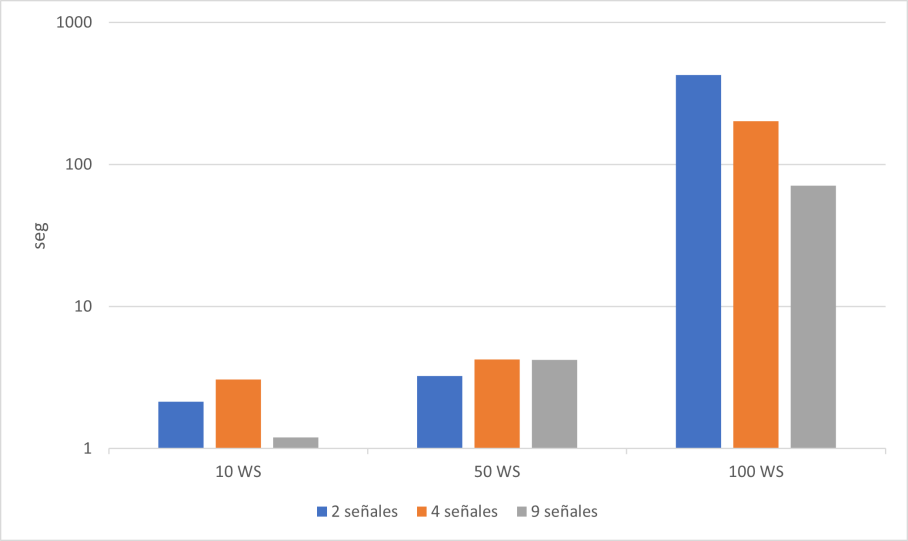


Figura 5.9: Relación del tiempo en reconocimiento PS.
(Escenario 2)

Los parámetros QoS pueden variar; sin embargo, hay casos en los que la similitud de los valores complica la elección de un servicio. Ar_WSDS, utilizando sus módulos de reconocimiento, puede analizar todos los parámetros y encontrar las diferencias entre sus valores, permitiendo así diferenciar un servicio de otros. La Tabla 5.7, muestra esta situación, en la que la similitud de los parámetros de la calidad del servicio para los servicios BINDService y DOTSGoePhone está en tres parámetros.

Las pruebas realizadas en el caso de estudio demostraron la eficacia del proceso de reconocimiento, en sus dos estrategias. La Tabla 5.6 muestra cómo fue necesario especificar los umbrales para el reconocimiento por claves, lo que significa que el desarrollador debe conocer de antemano los parámetros de calidad de los servicio deseados; sin embargo, el reconocimiento parcial ofrece la ventaja de analizar todos los WS y obtener los más adecuados, proporcionando una gama más amplia de SP en la selección de servicios.

La selección de un Δ para el servicio compuesto puede hacer que los resultados varíen, según sus SP (es decir, cuanto más alto sea el Δ, más restrictivo será el sistema a la hora de seleccionar sus servicios). La Figura 5.6 muestra cómo Ar_WSDS puede adaptar cualquiera de sus dos estrategias de reconocimiento para seleccionar servicios, siendo el reconocimiento por señales parciales el que ofrece los mejores resultados.

La tasa de éxito del reconocimiento varia conforme al incremento de señales, es decir, si un estrategia trabaja con 2 señales y otra con 7, el sistema tendrá una tasa de éxito mayor a la de menor cantidad de señales, debido a las bajas restricciones que el WS deberá cumplir, como se puede observar en la Figura 5.10. A medida que el cardinalidad de señales aumenta en un tipo de reconocimiento las tasas de éxito se van invirtiendo;no obstante, el reconocimiento SP obtuvo la mayor tasa, como ya ha mencionado esté tipo de estrategia es menos restrictiva y depende del valor promedio de los QoS procesados.

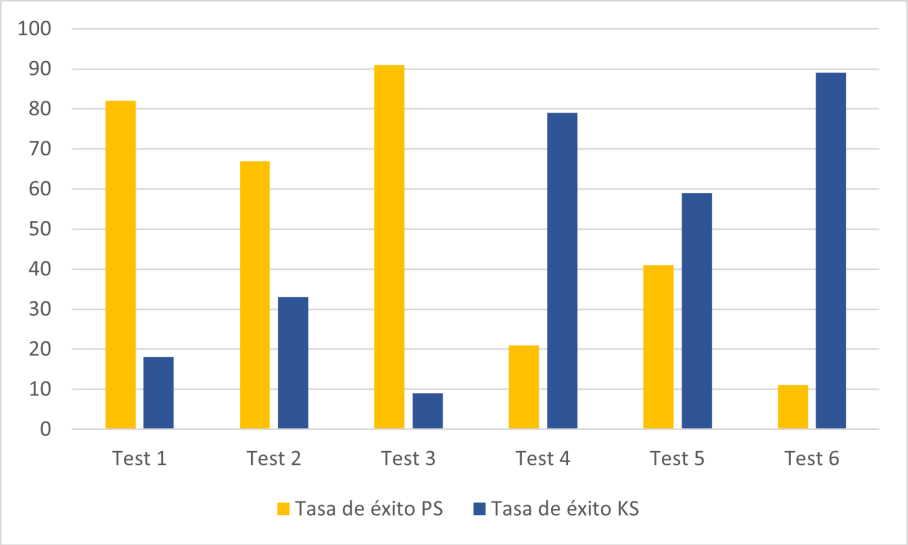


Figura 5.10: Tasa de éxito entre reconocimiento KS y PS.
(Escenario 3)

En el caso estudio de caso, se analizaron WS con un número variable de réplicas utilizando dos enfoques de selección -el primero definido por OAEQoS y el segundo por GDSA, ambos algoritmos utilizaron características de entrada símiles al enfoque propuesto. La composición fue desarrollada por un workflow en orden desde WS_1 hasta WS_9,permitiendo que el sistema tenga flexibilidad en la selección del servicio para en cada nodo del ACM.

Se realizaron pruebas usando un conjunto de servicios seleccionados a-priori que representaban los servicios esperados para un ACM en particular, se comparó el enfoque propuesto. La Tabla 5.9 muestra cómo Ar_WSDS consiguió los servicios esperados en sus dos estrategias de reconocimiento, en comparación con los otros dos enfoques; sin embargo, hubo dos casos en los que nuestro enfoque reconoció otros servicios. Está situación se produjo porque el sistema analizó todas las posibilidades de selección, y el resultado de la comparación se realizó sobre un conjunto predefinido a partir de la experiencia de un usuario.

La Figura 5.11 demuestra la eficacia del enfoque Ar_WSDS, tomando como punto de partida que GDSA produce más errores en el proceso de selección, ya que su estrategia de solución realiza búsquedas genéricas sin considerar ninguna estrategia clara de selección. Por lo tanto, es esencial que el proceso de selección de servicios cuente con estrategias de solución que exploren el análisis profundo de los parámetros de la calidad del servicio teniendo en cuenta su naturaleza,en este punto, GDSA no toman en cuenta la similitud de los RNF.

OAEQoS produjo resultados aceptables, y su capacidad para actualizar los parámetros de calidad de servicio es una característica interesante de estudiar, no obstante, la incorporación de estos procedimientos aumenta considerablemente el rendimiento del sistema. Esté enfoque requiere la incorporación de rutinas adicionales para analizar más parámetros de calidad,de igual forma, algunos valores QoS no son analizados apropiadamente cuando están sobre el límite de su condición de calidad. Por el contrario, Ar_WSDS, puede crear dinámicamente módulos

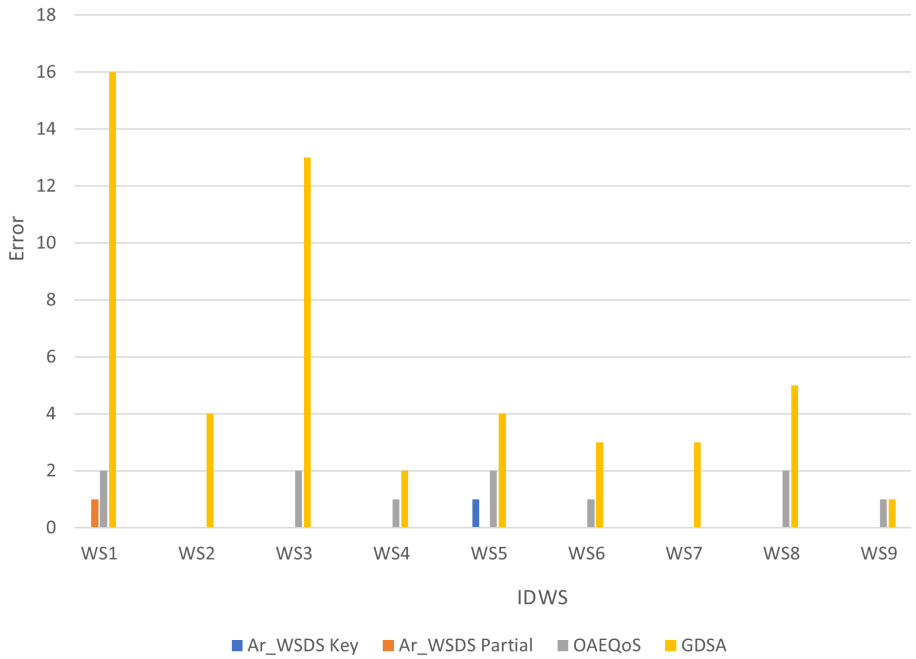


Figura 5.11: Tasa de error en el proceso de selección usando Ar_WSDS, OAEQoS y GDSA. (Escenario 5)

de reconocimiento para cualquier parámetro de calidad del servicio y evaluar sus umbrales en función del alcance Δ del servicio compuesto, permite al desarrollador obtener espectro mayor de servicios a través de la variabilidad de sus umbrales de calidad a nivel atómico y molecular.

La Tabla 5.11 muestra los resultados de la clasificación de los WS, en ella se puede observar que las dos estrategias de Ar_WSDS ofrecen una selección más adecuada, en comparación con los otros dos enfoques. Una ligera diferencia de clasificación existió en el reconocimiento clave frente al reconocimiento parcial, ya que este último tenía un espacio de búsqueda mayor. OAEQoS presentó diferencias bajo un mayor número de WS ya que, en su proceso de selección, debe elegir más servicios, excluyendo algunos del reconocimiento. GDSA, aunque su proceso de selección es más extenso que el de los otros enfoques, mostró diferencias significativas en su proceso de clasificación, en comparación con Ar_WSDS. Este enfoque seleccionó la gran mayoría de los servicios cuando sus parámetros de calidad de servicio tenían poca variabilidad.

Finalmente, la Tabla 5.12 muestra como afecta cada parámetro QoS de forma atómica en el momento de ser usado ese servicio en una ACM. Los criterios de Tiempo de Respuesta (P_1),Disponibilidad (P_2),Rendimiento (P_3), Tasa de éxito (P_4) y Documentación (P_9) , ofrecen una gama de posibilidades donde el servicio puede seleccionarse con un mayor rango de posibilidades dependiendo de la estrategia de que el desarrollador utilice (KS o bien PS), dando el mayor número de WS que podrían pertenecer al conjunto WS_C; no obstante, para el caso de los criterios Fiabilidad (P_5), Cumplimento (P_6),Latencia (P_7) y Mejores prácticas (P_8), se limita a tener WS de ranking 1, esto hace que si esos criterios son escogidos como restricciones del ACM , la cardinalidad del conjunto de los SP sea vea afectada , llevando a una reducción considerable

de WS que pueden aportar a ese nodo del modelo.

En estos casos, el desarrollador debe prever que criterios son los que necesita con mayor prioridad, ya que a medida que las restricciones aumenten el proceso de selección irá reduciendo es espacio de búsqueda y por ende las rutas de solución (SP).

CAPÍTULO 6

Conclusiones

La estrategia de selección de SW desarrollada ofrece un modelo dinámico para el reconocimiento de servicios para múltiples condiciones QoS. El modelo hace que la implementación de los módulos a nivel de desarrollo de software sea rápida y fácilmente escalable con la utilización de cualquier estrategia de optimización que pudiese ser aplicada sobre cada modulo reconocedor. La característica del funcionamiento del modelo biológico del neocórtex ofrece una forma de dividir el problema de selección de composición en tareas funcionales y descriptivas, cuya concepción de solución puede ser adaptada a otros modelos de búsqueda .

El descubrimiento y la selección de servicios son tareas complejas aún en la actualidad, debido a la disponibilidad de recursos en la etapa de diseño y de ejecución en composiciones. La investigación relativa a la composición de servicios se enfrenta a los retos que subyacen de la idea básica de cómo buscar y seleccionar un servicio de forma eficiente. El problema se agudiza en entornos cloud donde los recursos pueden estar o no disponibles en un tiempo determinado, por lo que un modelo de composición puede cambiar desde la fase de diseño hasta su ejecución. Es incipiente la recolección de datos a nivel de los criterios QoS con los que un SW es expuesto por sus proveedores, esto limita en gran medida una prueba real en contextos de despliegue directos de un servicio.

Es desarrollado un sistema de descubrimiento y selección de SW basado en el funcionamiento sistémico del cerebro considerando patrones. Su proceso utiliza dos estrategias: señales clave y parciales. Esta característica de reconocimiento ofrece un método para dividir el problema en trabajos más sencillos. Ar_WSDS cambia la condición de las señales clave y parciales para ajustarse a las características de un SW y ajusta su concepto en el ámbito de la QoS para el caso del reconocimiento de componentes software.

La propuesta de solución, puede reconocer múltiples patrones y dividirlos en sub-patrones (criterios de QoS) para ser reconocidos por la estrategia seleccionada por el desarrollador de software. Ar_WSDS se adapta dinámicamente al tipo de reconocimiento para cada patrón, y el sistema selecciona los servicios establecidos tanto en el ACM como para cada WS (Δ), el sistema es capaz de reconocer WS si bien por señales claves o bien por señales parciales, está última cuando el primer proceso no logra encontrar servicios candidatos.

Como contribución importante, se destaca que Ar_WSDS puede reconocer un conjunto de patrones que satisfacen condiciones de calidad dinámicas. La creación de módulos de reconocimiento dinámicos, basados en los RNF, proporciona una ventaja esencial para los desarrolladores que deseen utilizar este enfoque, ya que pueden personalizar sus búsquedas en función del requisito deseado o refinarlas para obtener mayores rutas de solución. El principio de funcionamiento del Ar_WSDS puede servir como base para otros trabajos que incorporen la noción de clasificación por QoS.

Es formalizado los SW y el modelo de composición desde una perspectiva matemática, permitiendo la interpretación y sistematización del sistema en una arquitectura modular, facilitando así su escalabilidad para la adición de nuevos parámetros de QoS. La QoS tiene una naturaleza dinámica de valores debido a la información suministrada por los proveedores, por lo que Ar_WSDS incluye un módulo para ensamblar los nuevos datos sin necesidad de alterar su programación base.

Según las pruebas realizadas, el reconocimiento KS proporcionó un mejor rendimiento que el reconocimiento PS, ya que el primero limita el número de SW_c y, por tanto, el espacio de búsqueda es mucho menor. Sin embargo, la estrategia PS fue más versátil a la hora de seleccionar los servicios, ya que ofrece una gama mucho más amplia de SP. La tasa de éxito en el reconocimiento varió en función del número de restricciones impuestas al modelo y del umbral de calidad impuesto a la composición. El rendimiento del sistema puede verse afectado cuando existen similitudes en los valores QoS de cada SW, ya que el sistema tendría que seleccionar todos esos servicios dentro del SP, y en este caso, el reconocimiento KS y PS podría generar los mismos resultados.

Según la evaluación, Ar_WSDS logró excelentes resultados en el descubrimiento y la selección de SW, en comparación con otros enfoques existentes. Su estrategia permite a los usuarios añadir más reconocedores interpretando rápidamente el parámetro de calidad de servicio requerido. En general, el enfoque propuesto es mucho más eficaz en la selección y clasificación. En particular, permite analizar cada valor de QoS de forma independiente, y como estos servicios pueden ser seleccionados para alcanzar los umbrales de calidad definido en ACM. Como el sistema no depende de los datos, los módulos implementados para cada tarea de reconocimiento son fácilmente escalables a otros sistemas que requieran la clasificación de elementos por algún atributo de calidad.

Finalmente,esté trabajo formula tres preguntas de investigación cuyo discernimiento se da a continuación:

¿Cuáles son los problemas específicos que enfrenta una composición de servicios en su etapa de búsqueda y selección de WS cuando operan con restricciones QoS?

La composición de servicios se ve enfrentada a problemas de optimización debido al crecimiento exponencial de servicios que son desplegados en diferentes proveedores con similitud de criterios QoS. En la sección 3.1, es realizado un análisis en detalle de los problemas existentes en la fase de búsqueda y selección. Los enfoques estudiados destacan la necesidad de establecer estrategias computacionales para analizar la singularidad en el comportamiento dinámico de

los QoS, en la identificación de los criterios que afectan directamente a la composición y cuales otros pueden ser secundarios. Así mismo, la desactualización de los métricas en los proveedores orientadas a sus QoS hacen que la tarea de selección sea aún incipiente.

Desde el punto de vista de complejidad computacional, la tarea de seleccionar WS puede convertirse un problema combinatorio, debido a que un WS requerido en una composición puede tener muchos servicios que satisfagan ese requerimiento; de ahí la importancia de analizar efectivamente los QoS a nivel de sus intervalos de calidad como de la naturaleza y/o atributo que representan. Ar_WSDS permite a través de sus estrategias de reconocimiento, analizar cada criterio tomando en cuenta su naturaleza y sus valores de calidad dentro de intervalos de aceptación y como estos aportan a una métrica de calidad del servicio compuesto.

¿Cuál sería la estrategia de solución para buscar y seleccionar WS bajo condiciones QoS que satisfaga los requerimientos no funcionales de una composición de servicios?

Esté trabajo propone un singular método de selección de WS en el área del reconocimiento de patrones que incorpora el concepto de *Reconocimiento de software*, presentado en la sección 4.2. Si bien el área de reconocimiento de patrones es muy amplia, sus principales usos se derivan de dominios sobre la ingeniería y afines con el objetivo de identificar entidades físicas o abstractas; acorde con está definición genérica se explora una novedosa estrategia de selección basado en el reconocimiento de componentes software,específicamente en el campo de los WS.

La estrategia desarrollada incorpora el análisis de requerimientos no funcionales sobre las dos entidades de una composición: el servicio atómico y el servicio molecular(compuesto), bajo la premisa de descubrir y seleccionar los servicios más óptimos. En función de esto, se crea Ar_WSDS, que permite un sistema que descubre y selecciona WS convirtiendo la composición y los servicios de un repositorio, en patrones y señales a ser procesados por módulos que seleccionan y categorizan un servicio.

A diferencia de los enfoques estudiados, Ar_WSDS opera con nueve criterios QoS los cuales pueden ser personalizados a nivel atómico para cada servicio presente en la composición (especificados en la Tabla 5.1). La versatilidad del enfoque propuesto, dota de resultados variables a partir de una métrica de calidad global asociada al servicio molecular. Adicionalmente, al descomponerse el problema de selección en módulos funcionales proporciona una manera simple y eficaz de análisis de WS y cómo deben ser evaluados para que se consideren los servicios óptimos y el modelo de composición logre su objetivo.

¿Qué análisis se tiene que llevar a cabo a nivel de QoS para categorizar un WS como el más óptimo para una composición de servicios?

Ar_WSDS integra la concepción de índices sobre una estructura de datos otorgándole significancia a esté valor para convertirlo en un ranking. El análisis se lleva a cabo con las estrategias de reconocimiento y consiste en dos cálculo de ranking, el primero orientado a cada QoS definido en los RNF y el segundo al servicio. Ambas estrategias utilizan como soporte un árbol binario de búsqueda, estructura escogida por su facilidad de implementación y un bajo costo computacional en procesos de inserción y búsqueda. Así mismo, se establece una categorización

para las rutas de solución que dependen del umbral de calidad que se desea alcanzar por el servicio compuesto.

Como propuesta de trabajos futuros y mejoras al sistema, es necesario implementar un módulo de reconocimiento que evalúe el estado de un servicio a seleccionar y descarte aquellos que dejaron de funcionar en los proveedores en tiempo real. adicional, es esencial crear módulos de software para mantener los conjuntos de datos actualizados en el repositorio base.

Para aumentar la eficacia del reconocimiento, se podría crear un modelo que permita crear un registro de los servicios más utilizados por los clientes; así, el sistema no necesitará realizar los procesos de búsqueda iniciales y sólo los realizará cuando nuevos servicios sean requeridos. Por último, se puede implementar un módulo de reconocimiento que analice el flujo de la composición para determinar la redundancia de operaciones, lo que permitiría afinar la composición y reducir el número de servicios.

CAPÍTULO 7

Anexos

Results response time

Rank	Response Time
1	111.67
2	113.0
3	114.0
4	114.67
5	115.0

Figura 7.1: Ranking de GoogleSearchService para Criterio P_1

Rank	availability
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
2	99.0
3	97.0
3	97.0
3	97.0
3	97.0
3	97.0
3	97.0
4	96.0
5	95.0

Figura 7.2: Ranking de GoogleSearchService para Criterio P_2

Rank	throughput
1	11.3
1	11.3
1	11.3
2	11.1
3	11.0
4	10.8
5	10.1

Figura 7.3: Ranking de GoogleSearchService para Criterio P_3

Rank	successability
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
1	100.0
2	99.0
2	99.0
2	99.0
2	99.0
2	99.0
2	99.0
2	99.0
3	98.0
3	98.0
3	98.0
3	98.0
3	98.0
3	98.0
3	98.0
4	97.0
4	97.0
4	97.0
4	97.0
4	97.0
4	97.0
4	97.0
4	97.0
5	96.0
5	96.0

Figura 7.4: Ranking de GoogleSearchService para Criterio P_4

Rank	reliability
1	73.0
1	73.0
1	73.0

Figura 7.5: Ranking de GoogleSearchService para Criterio P_5

Rank	compliance
1	78.0
1	78.0

Figura 7.6: Ranking de GoogleSearchService para Criterio P_6

Rank	compliance
1	78.0
1	78.0

Figura 7.7: Ranking de GoogleSearchService para Criterio P_7

Rank	best practices
1	84.0
1	84.0

Figura 7.8: Ranking de GoogleSearchService para Criterio P_8

Rank	documentation
1	12.0
1	12.0
1	12.0
1	12.0
1	12.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
2	10.0
3	9.0
3	9.0
3	9.0
3	9.0
3	9.0
4	8.0
4	8.0
4	8.0
4	8.0
4	8.0

Figura 7.9: Ranking de GoogleSearchService para Criterio P_9

Bibliografía

[1] S. Wang y col., «Cloud model for service selection», en *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, IEEE, 2011, págs. 666-671.

[2] M. Rathore y U. Suman, «Evaluating QoS parameters for ranking Web service», en *2013 3rd IEEE International Advance Computing Conference (IACC)*, 2013, págs. 1437-1442. DOI: 10.1109/IAdCC.2013.6514438.

[3] G. Spezzano, «Using Service Clustering and Self-Adaptive MOPSO-CD for QoS-Aware Cloud Service Selection», *Procedia Computer Science*, vol. 83, n.º Ant, págs. 512-519, 2016, ISSN: 18770509. DOI: 10.1016/j.procs.2016.04.245. dirección: http://dx.doi.org/10.1016/j.procs.2016.04.245.

[4] A. Ramirez y col., «Evolutionary composition of QoS-aware web services: a many-objective perspective», *Expert Systems with Applications*, vol. 72, págs. 357-370, 2017.

[5] D. Wang y col., «QoS and SLA Aware Web Service Composition in Cloud Environment.», en *KSII Transactions on Internet Information Systems*, vol. 10, 2016.

[6] J. Qi y col., «Knowledge based differential evolution for cloud computing service composition», *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, n.º 3, págs. 565-574, 2018, ISSN: 18685145. DOI: 10.1007/s12652-016-0445-5.

[7] H. Nacer, K. Beghdad Bey y N. Djebari, «Migration from web services to cloud services», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10542 LNCS, págs. 179-192, 2017, ISSN: 16113349. DOI: 10.1007/978-3-319-68179-5{_}16.

[8] E. Al-Masri y Q. H. Mahmoud, «Investigating web services on the world wide web», en *Proceedings of the 17th international conference on World Wide Web*, 2008, págs. 795-804.

[9] Y. Chen y A. Abhari, «An Agent-based Framework for Dynamic Web Service Selection», en *Proceedings of the 2008 Spring Simulation Multiconference*, ép. SpringSim '08, San Diego, CA, USA: Society for Computer Simulation International, 2008, 6:1-6:6, ISBN: 1-56555-319-5. dirección: http://dl.acm.org/citation.cfm?id=1400549.1400694.

[10] S. K. Garg, S. Versteeg y R. Buyya, «A framework for ranking of cloud computing services», *Future Generation Computer Systems*, vol. 29, n.º 4, págs. 1012-1023, 2012, ISSN: 0167739X. DOI: 10.1016/j.future.2012.06.006. dirección: http://dx.doi.org/10.1016/j.future.2012.06.006.

[11] Y. Jadeja y K. Modi, «Cloud computing-concepts, architecture and challenges», en *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, IEEE, 2012, págs. 877-880.

[12] P. Mell y T. Grance, «The NIST definition of cloud computing», *NIST Special Publication*, vol. 145, pág. 7, 2011, ISSN: 00845612. DOI: 10.1136/emj.2010.096966. arXiv: 2305-0543. dirección: http://www.mendeley.com/research/the-nist-definition-about-cloud-computing/.

[13] Z.-H. Zhan y col., «Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches», *ACM Computing Surveys*, vol. 47, n.º 4, págs. 1-33, 2015, ISSN: 03600300. DOI: 10.1145/2788397. dirección: http://dl.acm.org/citation.cfm?doid=2775083.2788397.

[14] L. Joyanes Aguilar, «COMPUTACIÓN EN LA NUBE. Notas para una estrategia española en cloud computing», *Revista del Instituto Español de Estudios Estratégicos*, págs. 89-112, 2012, ISSN: 2255-3479. dirección: http://revista.ieee.es/index.php/ieee/article/viewFile/10/8.

[15] D. Puthal y col., «Cloud computing features, issues, and challenges: a big picture. In Computational Intelligence and Networks», *Computational Intelligence and Networks (CINE), 2015 International Conference*, n.º Cine, págs. 116-123. 2015.

[16] T. Erl, R. Puttini y Z. Mahmood, *Cloud Computing: Concepts, Technology and Design*. United States: Prentice Hall PTR, 2013.

[17] M. J. Kavis, *Architecting the cloud: Design decisions for cloud computing service models (SaaS, PaaS, AND IaaS)*. United States: John Wiley Sons, 2014.

[18] J. Mestas, *Modelos de Despliegue en la Nube*, 2011. dirección: http://geekswithblogs.net/gotchas/archive/2011/10/03/modelos-de-despliegue-en-la-nube.aspx.

[19] A. Keshavarzi, A. T. Haghighat y M. Bohlouli, «Research challenges and prospective business impacts of cloud computing: A survey», *Proceedings of the 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS 2013*, vol. 2, págs. 731-736, 2013. DOI: 10.1109/IDAACS.2013.6663021.

[20] A. Alcocer, *CLOUD COMPUTING. TIPOS DE NUBES*, 2009. dirección: http://www.societic.com/2010/06/cloud-computing-tipos-de-nubes-de-aplicaciones/.

[21] A. B. Nassif y M. A. M. Capretz, «Offering SaaS as SOA Services», en *Innovations and Advances in Computer, Information, Systems Sciences, and Engineering*, Springer, 2013, págs. 405-414.

[22] F. Bocchio, «Estudio Comparativo De Plataformas Cloud Computing Para Arquitecturas Soa», Tesis doct., UNIVERSIDAD TECNÓLOGICA NACIONAL DE BUENOS AIRES, 2013.

[23] A. Wahab y T. R. Soomro, «Implementation Of Service Oriented Architecture Using ITIL Best Practices», *Journal of Engineering Science and Technology*, vol. 10, n.º 6, págs. 765-770, 2015.

[24] M. S. G. Sawant y S. T. Singh, «A Survey on Strategy-Proof Pricing for Cloud Service Composition», *Imperial Journal of Interdisciplinary Research*, vol. 3, n.º 1, 2016.

[25] K. M. Dhara, M. Dharmala y C. K. Sharma, «A Survey Paper on Service Oriented Architecture Approach and Modern Web Services», *All Capstone Project*, vol. 157, 2015.

[26] L.-J. Zhang y Q. Zhou, «CCOA: Cloud computing open architecture», en *Web Services, 2009. ICWS 2009. IEEE International Conference on*, Ieee, 2009, págs. 607-616.

[27] W.-T. Tsai, X. Sun y J. Balasooriya, «Service-oriented cloud computing architecture», en *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, IEEE, 2010, págs. 684-689.

[28] T. Erl, *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR, 2004.

[29] D. K Barry, *Service Architecture*, 2017. dirección: http://www.service-architecture.com/articles/web-services/web_services_explained.html (visitado 12-02-2017).

[30] IBM, *Conceptos de los servicios web JSON*, 2013. dirección: https://www.ibm.com/support/knowledgecenter/es/SSGMCP_5.1.0/com.ibm.cics.ts.mobileextensions.doc/concepts/concepts_json.html (visitado 12-02-2017).

[31] F. Belqasmi y col., «Soap-based vs. restful web services: A case study for multimedia conferencing», *IEEE internet computing*, vol. 16, n.º 4, págs. 54-63, 2012.

[32] A. W. Mohamed y A. M. Zeki, «Web services SOAP optimization techniques», en *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, IEEE, nov. de 2017, págs. 1-5, ISBN: 978-1-5386-2106-6. DOI: 10.1109/ICETAS.2017.8277881. dirección: http://ieeexplore.ieee.org/document/8277881/.

[33] S. Mumbaikar, P. Padiya y col., «Web services based on soap and rest principles», *International Journal of Scientific and Research Publications*, vol. 3, n.º 5, 2013.

[34] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. .º'Reilly Media, Inc.", 2011.

[35] T. Erl y col., *SOA with REST: Principles, Patterns Constraints for Building Enterprise Solutions with REST*. Prentice Hall Press, 2012.

[36] W3C Working Group, *Web Services Architecture*, 2004. dirección: https://www.w3.org/TR/ws-arch/#wsd.

[37] O. Dospinescu y M. Perca, «Web Services in Mobile Applications», *Informática Economica*, vol. 17, n.º 2/2013, págs. 17-26, 2013, ISSN: 14531305. DOI: 10.12948/issn14531305/17.2.2013.02.

[38] Z. Zheng, Y. Zhang y M. R. Lyu, «Investigating QoS of real-world web services», *IEEE Transactions on Services Computing*, vol. 7, n.º 1, págs. 32-39, 2014.

[39] R. Khadka y B. Sapkota, «An Evaluation of Dynamic Web Service Composition Approaches», en *4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, Athens: SciTePress, 2010, págs. 67-79. dirección: http://doc.utwente.nl/72798/1/Khadka_ACT4SOC_2010.pdf.

[40] D. A. D'Mello, V. S. Ananthanarayana y S. Salian, «A review of dynamic web service composition techniques», *Springer*, págs. 85-97, 2011.

[41] M. Champion y col., «Web services architecture», *W3C working draft*, vol. 14, 2002.

[42] I. Weber, H. Y. Paik y B. Benatallah, «Forms-based service composition», *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7084 LNCS, págs. 627-635, 2011, ISSN: 03029743. DOI: 10.1007/978-3-642-25535-9_49.

[43] M. Moghaddam y J. G. Davis, «Service selection in web service composition: A comparative review of existing approaches», en *Web Services Foundations*, Springer, 2014, págs. 321-346.

[44] L. J. White y col., «Maintenance of service oriented architecture composite applications: static and dynamic support», *Journal of Software: Evolution and Process*, vol. 25, n.º 1, págs. 97-109, 2013.

[45] P. Bartalos y M. Bieliková, «Automatic dynamic web service composition: A survey and problem formalization», *Computing and Informatics*, vol. 30, n.º 4, págs. 793-827, 2012.

[46] D. E. Bakken, *Middlewares*, 2005. dirección: http://www.dia.uniroma3.it/$%5Csim$cabibbo/ids/altrui/middleware-bakken.pdf.

[47] A. Abdalahem, P. Further y J. Pasquier, *Case Study of Web services Composition-Department of Informatics at the University of Fribourg*, 2009. dirección: https://diuf.unifr.ch/drupal/softeng/sites/diuf.unifr.ch.drupal.softeng/files/file/publications/internal/WP09-03.pdf.

[48] J. A. Cubillos Patiño y J. E. Burbano Sandoval, «DESCRIPCIÓN, LOCALIZACIÓN Y COMPOSICIÓN SEMÁNTICA DE SERVICIOS WEB COMO SOLUCIÓN PARA EL SISTEMA DE GESTIÓN DE CUENTAS DE USUARIO DE LA RED DE DATOS DE LA UNIVERSIDAD DEL CAUCA», Cali, inf. téc., 2005, pág. 246.

[49] L. Wu, S. K. Garg y R. Buyya, «SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments», en *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE Computer Society, 2011, págs. 195-204.

[50] M. Alhamad, T. Dillon y E. Chang, «Sla-based trust model for cloud computing», en *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, Ieee, 2010, págs. 321-324.

[51] L. Wang y col., «Cloud computing: a perspective study», *New Generation Computing*, vol. 28, n.º 2, págs. 137-146, 2010.

[52] H. Mezni y M. Sellami, «Multi-cloud service composition using Formal Concept Analysis», *Journal of Systems and Software*, vol. 134, págs. 138-152, 2017, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.08.016. dirección: http://www.sciencedirect.com/science/article/pii/S0164121217301760.

[53] S. Sundareswaran, A. Squicciarini y D. Lin, «A brokerage-based approach for cloud service selection», en *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, IEEE, Honolulu, 2012, págs. 558-565.

[54] L. Sun y col., «Cloud service selection: State-of-the-art and future research directions», *Journal of Network*, 2014. dirección: http://www.sciencedirect.com/science/article/pii/S108480451400160X.

[55] Q. Z. Sheng y col., «Web services composition: A decade's overview», *Information Sciences*, vol. 280, págs. 218-238, 2014, ISSN: 00200255. DOI: 10.1016/j.ins.2014.04.054.

[56] C. N. Höfer y G. Karagiannis, «Cloud computing services: taxonomy and comparison», *Journal of Internet Services and Applications*, vol. 2, n.º 2, págs. 81-94, 2011.

[57] J. Sidhu y S. Singh, «Improved topsis method based trust evaluation framework for determining trustworthiness of cloud service providers», *Journal of Grid Computing*, vol. 15, n.º 1, págs. 81-105, 2017.

[58] J. Upadhyaya y N. J. Ahuja, «Quality of service in cloud computing in higher education: A critical survey and innovative model», en *I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2017 International Conference on*, IEEE, 2017, págs. 137-140.

[59] F. M. Mena y col., «Web service composition using the bidirectional Dijkstra algorithm», *IEEE Latin America Transactions*, vol. 14, n.º 5, págs. 2522-2528, 2016.

[60] M. Alrifai, T. Risse y W. Nejdl, «A hybrid approach for efficient Web service composition with end-to-end QoS constraints», *ACM Transactions on the Web (TWEB)*, vol. 6, n.º 2, pág. 7, 2012.

[61] H. T. Dinh y col., «A survey of mobile cloud computing: architecture, applications, and approaches», *Wireless communications and mobile computing*, vol. 13, n.º 18, págs. 1587-1611, 2013.

[62] M. Vera, «Modelo para evaluar la gestión de sistemas de información en las entidades públicas colombianas», Tesis doct., Universidad Nacional de Colombia, 2013, pág. 226. dirección: https://repositorio.unal.edu.co/handle/unal/21619%20http://www.bdigital.unal.edu.co/12582/.

[63] R. L. Baskerville y M. D. Myers, «Fashion Waves in Information Systems Research and Practice», *MIS Quarterly*, vol. 33, n.º 4, págs. 647-662, oct. de 2009, ISSN: 02767783. DOI: 10.2307/20650319. dirección: http://www.jstor.org/stable/20650319.

[64] M. Ghobaei-Arani y col., «CSA-WSC: cuckoo search algorithm for web service composition in cloud environments», *Soft Computing*, vol. 22, n.º 24, págs. 8353-8378, 2018.

[65] B. Di Martino, G. Cretella y A. Esposito, «Cloud services composition through cloud patterns: a semantic-based approach», *Soft Computing*, vol. 21, n.º 16, págs. 4557-4570, 2017.

[66] S. Rangarajan, «Qos-Based Web Service Discovery And Selection Using Machine Learning», *EAI Endorsed Transactions on Scalable Information Systems*, vol. 5, n.º 17, jun. de 2018. DOI: 10.4108/eai.29-5-2018.154809.

[67] D. G. Chakravarthy y S. Kannimuthu, «Extreme Gradient Boost Classification Based Interesting User Patterns Discovery for Web Service Composition», *Mobile Networks and Applications*, vol. 24, n.º 6, págs. 1883-1895, 2019.

[68] J. Sha, Y. Du y L. Qi, «A user requirement oriented web service discovery approach based on logic and threshold petri net», *IEEE/CAA Journal of Automatica Sinica*, vol. 6, n.º 6, págs. 1528-1542, 2019. DOI: 10.1109/JAS.2019.1911657.

[69] Y. Wu y col., «A Multilevel Index Model to Expedite Web Service Discovery and Composition in Large-Scale Service Repositories», *IEEE Transactions on Services Computing*, vol. 9, n.º 3, págs. 330-342, 2016, ISSN: 19391374. DOI: 10.1109/TSC.2015.2398442.

[70] S. K. Garg, S. Versteeg y R. Buyya, «Smicloud: A framework for comparing and ranking cloud services», en *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, IEEE, 2011, págs. 210-218.

[71] R. Gomes y col., «A middleware-based approach for QoS-aware deployment of service choreography in the cloud», en *Proceedings of the 11th Middleware Doctoral Symposium*, ACM, 2014, pág. 2.

[72] A. Bentaleb y A. Ettalbi, «Toward Cloud SaaS for web service composition optimization based on genetic algortihm», en *Cloud Computing Technologies and Applications (CloudTech), 2016 2nd International Conference on*, IEEE, 2016, págs. 147-152.

[73] A. A. Wakrime y S. Jabbour, «Formal approach for QoS-aware cloud service composition», *Proceedings - 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2017*, págs. 30-35, 2017. DOI: 10.1109/WETICE.2017.61.

[74] H. Wada y col., «A Multiobjective Optimization Framework for SLA-Aware Service Composition», *Services Computing, IEEE Transactions on*, vol. 5, n.º 3, págs. 358-372, 2012.

[75] X. Zhou y F. Mao, «A semantics web service composition approach based on cloud computing», en *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, IEEE, 2012, págs. 807-810.

[76] R. O. Duda, P. E. Hart y D. G. Stork, *Pattern Classification*. California: Jhon Wiley & Sons, 1997.

[77] G. Chandrashekar y F. Sahin, «A survey on feature selection methods», *Computers and Electrical Engineering*, vol. 40, n.º 1, págs. 16-28, 2014, ISSN: 00457906. DOI: 10.1016/j.compeleceng.2013.11.024.

[78] S. Khalid, T. Khalil y S. Nasreen, «A Survey Of Feature Selection And Feature Extraction Techniques In Machine Learning», inf. téc., 2014. dirección: https://www.researchgate.net/publication/265727419.

[79] P. Oliveri y col., «Qualitative pattern recognition in chemistry: Theoretical background and practical guidelines», *Microchemical Journal*, vol. 162, pág. 105 725, 2021, ISSN: 0026-265X. DOI: https://doi.org/10.1016/j.microc.2020.105725. dirección: https://www.sciencedirect.com/science/article/pii/S0026265X20333336.

[80] C. H. Chen, *Handbook of Pattern Recognition and Computer Vision*, 5th, C. H. Chen, ed. World Scientific, 2016. DOI: 10.1142/9503.

[81] P. C. E. Gilberto, «A Recursive Pattern Recognition Algorithm», *Revista Técnica de la Facultad de Ingenier{\'\i}a Universidad del Zulia*, vol. 40, n.º 2, 2017. dirección: http://www.scielo.org.ve/scielo.php?pid=S0254-07702017000200005&script=sci_arttext&tlng=en.

[82] E. Puerto y J. Aguilar, «Extended abstract: Formal description of a pattern for a recursive process of recognition», en *2016 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2016 - Proceedings*, IEEE, ed., Cucuta: IEEE, nov. de 2016, págs. 1-2, ISBN: 9781509051052. DOI: 10.1109/LA-CCI.2016.7885746. dirección: http://ieeexplore.ieee.org/document/7885746/.

[83] R. H. Reussner y F. Plasil, *Component-Based Software Engineering*. Springer, 2010.

[84] J.-H. Jang, D.-H. Shin y K.-H. Lee, «Fast quality driven selection of composite Web services», en *2006 European Conference on Web Services (ECOWS'06)*, IEEE, 2006, págs. 87-98.

[85] E. M. Dashofy, A. v. d. Hoek y R. N. Taylor, «A comprehensive approach for the development of modular software architecture description languages», *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 14, n.º 2, págs. 199-245, 2005.

[86] S. Wang y col., «Cloud model for service selection», en *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, IEEE, 2011, págs. 666-671.

[87] M. Hasnain y col., «Benchmark dataset selection of Web services technologies: A factor analysis», *IEEE Access*, vol. 8, págs. 53 649-53 665, 2020.

[88] L. León, «Tejiendo Algoritmos», *Universidad de los Andes. Consejo de Publicaciones. Mérida*, 2012.

[89] Middleware Labs Inc, *What is Service Discovery? Complete Guide (Basic + Advance)*, 2021. dirección: https://middleware.io/blog/service-discovery/.

[90] S. Li y col., «Understanding and addressing quality attributes of microservices architecture: A Systematic literature review», *Information and Software Technology*, vol. 131,

pág. 106 449, 2021, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2020.106449. dirección: https://www.sciencedirect.com/science/article/pii/S0950584920301993.

[91] C. Richardson, *Microservices patterns: with examples in Java*. Simon y Schuster, 2018.