Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Departamento de Computación

Tesis:

Análisis en la correspondencia de tipos de datos para la similitud de servicios Web

Que para obtener el grado de Maestro en Ciencias de la Computación presenta:

Jorge Náder Roa

Director:

Dr. José Guadalupe Rodríguez García

Codirector:

Dra. Maricela Claudia Bravo Contreras

Resumen

Con el creciente desarrollo de servicios web (SW), es necesaria la elaboración de métodos y herramientas capaces de ofrecer clasificaciones y búsquedas eficientes de SW. En la actualidad los métodos de descubrimiento o búsqueda carecen de mecanismos robustos que otorguen resultados adecuados en la búsqueda de una determinada operación de un SW.

El proceso de descubrimiento de un SW es realizado mediante comparaciones entre los datos proporcionados por el usuario para la búsqueda y las especificaciones de las operaciones de los SW, de tal manera que el módulo principal durante el descubrimiento es el proceso de comparación.

Los trabajos realizados para mejorar los procesos de búsqueda, proponen metodologías que enriquecen las comparaciones, destacándose cuatro principales enfoques, cuyos elementos de comparación son: descripción semántica de operaciones, semántica proporcionada por el nombre de operaciones, flujos de trabajo y firmas de funciones. Siendo la comparación por firmas de funciones de mayor interés para éste trabajo, debido a que este enfoque no permite obtener grados de similitud en la comparación de operaciones, otorgando descubrimientos poco tolerantes a variaciones en la similitud de los parámetros de búsqueda.

En este proyecto se propone el desarrollo de una herramienta capaz de realizar comparaciones entre las operaciones de diferentes SW, ofreciendo grados de similitud como resultado. Las comparaciones serán abordadas representando los tipos de dato de entrada y salida en estructuras de dato que faciliten la correlación de dos operaciones.

Abstract

With the increasing development of web services (WS), it is necessary to develop methods and tools capable to provide clasifications and efficient WS searches. Nowadays methods of discovery and searching have a lack of strong mechanisms to give the best results searching a particular operation of a WS.

The discovery process of WS is done by comparisons between the data provided by the user to search for and specifications of WS operations, making the comparison process the main module on the discovery.

The related work to improve the search process, propose methodologies that enrich comparisons, highlighting four main approaches which elements of comparison are: semantic descriptions of operations, the semantic provided by the name of operations, workflows and signatures of functions. Making the signature comparison the most important aproach for this work, because it does not obtain similarity degrees in comparison operations, providing hard findings without degrees of similarity on the searched parameters.

In this project is proposed the development of a tool to make comparisons between different WS operations, offering degrees of similarity. Comparisons will be done by representing data types of input and output WS messages in structures that ease the correlation between two operations.

Agradecimientos

Agradezco a Dios y a mi familia por ser siempre un soporte en mi vida y por ayudarme a lograr un objetivo más.

Agradezco al Consejo Nacional De Ciencia Y Tecnología y al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional por brindarme la oportunidad de cursar un posgrado en su programa de maestrías.

Índice general

Capítulo 1

Introducción

Con la llegada de la Web 2.0, el auge de los *blogs*, el uso de CMS (*Content Management System*) y las redes sociales se fue incorporando naturalmente el uso de los Servicios Web(SW) para permitir que los servicios ofrecidos por un sitio web se propagen de mejor manera, al proporcionar interfaces fácilmente evocables por sistemas independientes con plataformas heterogéneas [1, 2].

En la actualidad resulta común e imperceptible para los usuarios que navegan por la red hacer uso de SW en operaciones frecuentes como: realizar compras y pagos por internet, descargar la información de su perfil de una red social, realizar una búsqueda de los videos más populares, verificar el tipo de cambio, etc. Cada una de ellas son acciones frecuentes que son encapsuladas en un SW para extender la versatilidad de los servicios ofrecidos por un sitio Web.

Con la proliferación de los SW fue necesario el desarrollo de técnicas y herramientas que permitieran la organización, selección y el fácil acceso a los SW [3, 4, 5, 6, 7, 8], resultando en modelos de descubrimiento como lo son: UDDI (*Universal Description Discovery and Integration*) [9] y WSIL (*Web Services Inspection Language*) [10]. Los modelos de descubrimiento otorgan al usuario la capacidad de realizar búsquedas sobre un conjunto de SW a través de métodos específicos para realizar filtros en la búsqueda de un SW [8].

Los métodos empleados para la búsqueda y descubrimiento de SW realizan comparaciones entre los elementos de las descripciones de los SW, identificando las descripciones con mayor relevancia para el proceso de búsqueda. Sin embargo, las especificaciones de los SW no proporcionan descripciones suficientes para obtener rastreos eficientes de SW, ya que sólo describen las interfaces de las operaciones implementadas y su método de invocación, pero no contienen descripciones textuales completas sobre las funcionalidades de las operaciones, razón por la cual no es posible obtener comparaciones cercanas al contexto de búsqueda [7, 5, 6, 1, 8].

Debido a la importancia de realizar comparaciones durante el proceso de descubrimiento de un

SW, muchos de los trabajos realizados otorgan soluciones que mejoran el proceso de comparación entre diferentes operaciones de SW, obteniendo un conjunto de operaciones cuyas similitudes están más apegadas al criterio de búsqueda.

1.1. Motivación

La composición de servicios es la creación de nuevos servicios, a partir de la combinación de SW elementales, los cuales son pequeños bloques funcionales con objetivos independientes, que en combinación resuelven un problema de mayor complejidad [11], brindando facilidad de manejo a los usuarios. No obstante, la composición de servicios denota el hecho de una inherente orquestación entre servicios elementales para alcanzar correctamente el objetivo del servicio compuesto.

La orquestación de un servicio compuesto combina varios SW dentro de un flujo de trabajo que posee diferentes rutas de ejecución. La naturaleza compleja de la orquestación de servicios radica en la capacidad de establecer la secuencia de ejecución entre las operaciones de los SW elementales, ya que es necesario conocer de manera exacta los parámetros de invocación y resolución de las mismas, para empatar adecuadamente los tipos de dato resueltos por cada operación y así establecer un flujo de ejecución correcto en la resolución de un problema.

Actualmente los SW pueden ser encontrados en repositorios públicos o privados, ya sea mediante servidores UDDI [9] o mercados de servicios (*marketplaces*) tales como XMethods [1], WebserviceX [2], Workday [3] y salesforce [4], los cuales proporcionan elementos de búsqueda simples para la localización de SW y no permiten realizar búsquedas avanzadas que incorporen semántica o estructuras de datos como parámetros específicos de localización [1]; por lo que aun no existen herramientas integradas a los servicios de descubrimiento que asistan al usuario en la selección de un servicio web de manera avanzada.

La composición automática de servicios Web requiere de comparaciones que demuestren la compatibilidad entre las estructuras de datos necesarias para la composición de un servicio [12, 13, 14, 4]; sin embargo, no hay una metodología que realice una comparación estructural de tipos de datos de las operaciones de un servicio Web.

Es posible integrar los algoritmos de comparación de SW al funcionamiento de los servidores de descubrimiento, extendiendo el API de interacción con el servidor.

[1] XMethods, Repositorio público de Servicios Web, http://www.xmethods.net/ Octubre 2013

[2] WebserviceX, Repositorio público de Servicios Web, http://www.webservicex.net/ Octubre 2013

[3] Workday, Repositorio público de Servicios Web, http://www.workday.com/ Octubre 2013

[4] salesforce, Repositorio público de Servicios Web, http://www.salesforce.com/ Octubre 2013

1.2. Objetivos del Trabajo

General

Definir una metodología para encontrar la correspondencia de tipos de datos entre operaciones de servicios Web.

Particulares

1. Extraer las definiciones de tipos de dato explícitas e importadas definidas en las especificaciones de los servicios web.

2. Definir una estructura para la representación de los tipos de dato requeridos en la entrada y salida de las operaciones de los servicios web.

3. Implementar un método reutilizable y adecuado para almacenar las estructuras de tipo de dato.

4. Definir un algoritmo que permita realizar la comparación y estimación de similitud entre dos estructuras de tipo de dato.

5. Desarrollar una aplicación Web que realice la comparación de similitud entre las operaciones de un determinado número de servicios web, que informe el grado de similitud entre las operaciones comparadas.

1.3. Estado del Arte

El descubrimiento y la composición automática de SW han recibido especial atención en el ámbito académico por la necesidad de desarrollar métodos eficientes en la organización, descubrimiento y utilización de SW como elementos individuales que componen a un servicio complejo.

Existen dos corrientes a través de las cuales es abordado el problema de descubrimiento de SW, el desarrollo de repositorios de especificaciones WSDL enriquecidas semánticamente y el descubrimiento a través de servicios interconectados que sirven de índice para ubicar nuevos SW, cuyas especificaciones sean iguales o parecidas entre ellos.

Sin importar la corriente de descubrimiento sobre la que se trabaje, ambas requieren de comparaciones entre SW, debido a que a través de ellas se evalúan las similitudes que permiten la clasificación de SW en categorías funcionales, ya sea en repositorios o para integrar índices de interconexión entre SW iguales o parecidos.

Los trabajos realizados para mejorar las comparaciones entre SW, fueron desarrollados con enfoques diferentes, los cuales difieren en los elementos tomados en cuenta para realizar comparaciones entre las especificaciones de los SW [2].

Aunque la mayoría ellos manejan el concepto de realizar búsquedas semánticas sobre las descripciones WSDL, cada uno proporciona una manera diferente de realizar la comparación y estimación de similitud entre las operaciones de un conjunto de SW.

Las vertientes más comunes de las comparaciones en la selección de los parámetros para la identificación y filtraje de SW son [2, 6, 8]:

1. Comparaciones basadas en la descripción semántica del servicio: las cuales obtienen la semántica relacionada con las descripciones de funcionalidad existentes en la especificación WSDL de un SW, para posteriormente clasificar a los SW en categorías de operación estableciendo relaciones de similitud entre los SW de cada categoría.

2. Comparaciones exactas entre los tipos de dato de las operaciones: esta estrategia realiza correlaciones entre las interfaces de las operaciones comparadas, para determinar si las declaraciones de las operaciones son idénticas o no.

3. Comparaciones semánticas por nombre de operación: clasifican las operaciones de los SW basándose en la interpretación semántica de los nombres asignados a las operaciones y estiman la similitud entre ellos, por la distancia semántica que poseen con la raíz o palabra origen del nombre de la operación.

4. Comparaciones de flujos de trabajo: el objetivo de esta técnica consiste en representar los flujos de trabajo de los SW como autómatas finitos deterministas y obtener todas las posibles cadenas aceptables por el autómata, de tal manera que si dos autómatas pertenecientes a diferentes SW procesan las mismas cadenas, se dice que los servicios son equivalentes, en su defecto se proporciona un grado de similitud basado en el número de cadenas idénticas que son aceptadas por ambos autómatas.

Eleni Stroulia y Yiqiao Wang [6], desarrollaron un conjunto de métodos que permiten realizar el descubrimiento de SW de manera programable, utilizando la información disponible en los archivos WSDL. Los métodos desarrollados utilizan como información de entrada las descripciones de los identificadores WSDL y la estructura de las operaciones del servicio web.

En general el trabajo describe dos categorías para la clasificación de métodos de descubrimiento en base al tipo de comparación que realizan, las cuales son: comparaciones por firma de operaciones y por especificación de elementos, la primera de ellas compara interfaces de operaciones y permite determinar si la especificación de las interfaces comparadas son idénticas o no; la segunda categoría

realiza una comparación semántica entre los textos descriptivos de los elementos de un servicio web a fin de obtener grados de similitud entre los temas que describen a un conjunto SW.

El método de descubrimiento desarrollado habilita la búsqueda de SW de manera programable al realizar comparaciones entre una descripción objetivo del servicio a identificar y las descripciones provistas por los identificadores WSDL de un conjunto de SW, donde la descripción objetivo es proporcionada por el usuario del servicio de descubrimiento, a través de texto simple en lenguaje natural. El desarrollo provisto hace uso de una de las clasificaciones antes mencionadas, realizando la comparación de especificación de elementos para identificar los servicios cuyos propósitos sean similares a los solicitados por el servicio de descubrimiento; adicionalmente, se usa un método extra para acotar con mayor precisión los SW que tengan mayor relación entre sí, para lo cual se realiza una comparación entre las estructuras de las operaciones de los SW, mapeando sobre una matriz todos los tipos de dato pertenecientes a las operaciones a comparar y asignando un peso a las intersecciones que resulten de tipos de datos idénticos, estableciendo un niveles de similitud entre los SW pertenecientes al conjunto acotado por la comparación de especificación de elementos, organizando en una lista los SW encontrados, colocando en las primeras posiciones los servicios con mayor nivel de similitud y en posiciones relegadas a los SW cuyos niveles de similitud sean bajos.

El trabajo hace uso del proyecto WordNet [5] el cual es una base de datos léxica inspirada en teorías psicolingüísticas sobre la memoria léxica de los humanos. Sustantivos, verbos y adjetivos del idioma inglés son organizados en conjuntos de sinónimos que representan conceptos léxicos.

WordNet es utilizado en este proyecto durante la etapa de comparación entre descripciones para determinar la similitud entre los conceptos léxicos que sean encontrados en las etiquetas WSDL.

Mehmet Snevar y Ayse Bener [7], proponen un algoritmo de matchmaking y una arquitectura extensible basada en servicios de descubrimiento y arquitecturas de la web semántica para facilitar la búsqueda y localización de SW.

La arquitectura propuesta, intenta ser una alternativa semántica a los servicios de descubrimiento comunes, enriqueciendo el contenido semántico de las descripciones de los SW y utilizando un algoritmo para relacionar e identificar similitudes entre las operaciones de los mismos. La similitud entre SW es estimada a través de un algoritmo de matchmaking, el cual utiliza como parámetro de similitud la distancia semántica existente entre dos palabras, de tal manera que los términos comparados en el algoritmo denoten el grado de similitud existente entre ellos, por la distancia terminológica que existe entre ambos términos.

El algoritmo, busca similitudes entre los nombres asignados a los mensajes de entrada y salida de cada operación a comparar, con la finalidad de estimar si la semántica provista por cada palabra

[5] WordNet, Base de datos léxica en idioma inglés, http://wordnet.princeton.edu/
Octubre 2013

pertenece al mismo contexto y así determinar qué tan parecidas son dos operaciones cualesquiera, para ello establecen y asignan niveles fijos de semejanza dependiendo el grado de similitud semántico que posean dos términos cualesquiera, ya sea "Exacto", "Plug-in", "Subsumido" o "Fallido"; adicionalmente, asignan un peso para expresar la distancia semántica entre los términos comparados, el cual se encuentra en el rango [0,1].

Para mantener la extensibilidad de la arquitectura, el algoritmo de matchmaking realiza la búsqueda de las distancias semánticas entre los individuos de una ontología de tipo OWL-S la cual permite el descubrimiento de servicios de manera semántica, a través de la terminología disponible en la ontología.

Akin Günay y Pinar Yolum [15], presentan una técnica semántica de matchmaking para SW basada en el proceso interno de los servicios. La propuesta combina heurísticas de similitud estructural, máquinas de estado finito y métricas de similitud semántica basadas en ontologías para estimar semejanzas entre servicios.

Para determinar la similitud de dos servicios, el algoritmo requiere serializar tres etapas que componen al matchmaking, inicialmente se utilizan las heurísticas estructurales para comparar individualmente los procesos atómicos que componen el flujo de ejecución de un servicio compuesto y los procesos atómicos del servicio a comparar. En esta comparación, si dos procesos atómicos son idénticos, las heurísticas estructurales asignan un valor de similitud de 1 y 0 en otro caso, donde este enfoque restringe únicamente a un proceso de matching exacto. Adicionalmente, es necesario el descubrimiento de la similitud parcial que se encuentra entre dos SW, para ello se relajan los umbrales de similitud entre procesos diferentes para obtener un valor en el rango [0,1].

La técnica de matchamking propuesta, requiere que los SW a comparar sean modelados previamente en máquinas de estado finito (MEF) por los desarrolladores, lo cual es requerido en la segunda etapa del algoritmo, misma que compara dos MEF, generando todas las posibles secuencias de los lenguajes aceptados por ambas máquinas y comparándolas entre sí, para obtener valores de similitud entre las secuencias similares. Para calcular los valores de similitud estructurales obtenidos en la segunda etapa, el algoritmo hace uso de las siguientes cuatro heurísticas: conteo común de procesos, sub-cadena común más grande, sub-secuencia común más grande y distancia editable, las cuales permiten valorar el tipo de parecido en las secuencias de las MEF.

Finalmente, la tercera etapa realiza una comparación semántica, la cual utiliza ontologías generales de dominios particulares que tienen relación con la terminología empleada en las definiciones de los SW. Para establecer una tasa de similitud entre los términos semánticos comparados, los autores desarrollaron una nueva métrica de similitud semántica llamada "Semantic cover rate" (SCR) para calcular la similitud entre dos términos, basados en ontologías de dominio general.

El resultado de las tres etapas presenta un conjunto de SW cuyos dominios y flujos de trabajo

son parecidos organizándolos por el nivel de similitud encontrado entre ellos.

Abdali Mohammadi et al. [3], Proponen un método para el enriquecimiento de directorios en un sistema de descubrimiento de SW, con la finalidad de otorgar búsquedas simples y precisas sobre un conjunto de SW previamente indexado y catalogado en base a las similitudes semánticas que los servicios posean.

El proyecto basa sus desarrollos en realizar mejoras al servicio de descubrimiento UDDI, el cual provee directorios organizados para la clasificación de especificaciones WSDL y un Framework de comunicación para establecer contacto con el o los repositorios disponibles en el servidor UDDI, a fin de realizar búsquedas categorizadas sobre el conjunto de especificaciones disponibles en el servidor de descubrimiento.

El método propuesto supone la existencia de un directorio de especificaciones de SW semánticamente enriquecido y cuyos servicios han sido clasificados mediante las similitudes existentes entre ellos. Haciendo alusión al concepto de redes sociales, donde los perfiles de diferentes individuos se encuentran relacionados, el sistema de enriquecimiento se basa en la creación de una red de recomendación entre SW, donde los servicios se recomiendan dependiendo de la similitud existente entre ellos. Las relaciones entre SW son descritas a través del lenguaje para la descripción de recursos RDF ya que mapea relaciones semánticas intrínsecas entre los SW y permite extender la información almacenada a través de la "Web of data".

El sistema de recomendación propuesto enriquece los directorios de especificaciones de servicios, mediante el uso de un grafo que representa todas las relaciones de similitud existentes entre los SW. El grafo de relaciones es mapeado fácilmente a una ontología gracias al formato RDF, permitiendo realizar inferencias sobre las relaciones y descubriendo conexiones implícitas entre los SW.

Finalmente, para realizar consultas al servicio de descubrimiento enriquecido, se hace uso de SPARQL para consultar las relaciones expresas en formato RDF y de esta manera localizar los SW que ajusten de mejor manera a los parámetros de búsqueda ingresados por el usuario.

Lin Lin y I. Budak Arpinar [5], presentan una técnica para el descubrimiento de relaciones semánticas entre las pre condiciones y post condiciones de diferentes servicios, haciendo uso de sus descripciones ontológicas. Esto permite identificar las relaciones complementarias existentes entre diferentes SW, revelando la compatibilidad entre ellos para realizar una composición automática de servicios.

La técnica propuesta, requiere como parámetro de entrada una descripción breve del servicio web a componer, para identificar el contexto de acción del servicio requerido y de esta manera buscar ontológicamente las conexiones semánticas entre las los nombres asignados a las operaciones de entrada y salida de diferentes SW. Esta búsqueda regresa un conjunto de SW con posibilidades de integración en un servicio compuesto, sin embargo, no es capaz de establecer el orden y tipo de

ejecución dentro del flujo de trabajo del servicio requerido, esto se debe a que no se conoce el tipo de relaciones existentes entre los SW previamente filtrados.

Para establecer el tipo de relación existente entre un conjunto de SW con posibilidades de composición, los autores identifican y establecen cuatro tipos de relaciones existentes entre los servicios atómicos que integran a un servicio compuesto, los cuales son: Relación de prerrequisito, de ejecución paralela, de sustitución y de inclusión. Al igual que el proceso de identificación de SW necesarios en una composición, la identificación de relaciones se realiza mediante el uso de ontologías que permiten obtener las relaciones entre los SW candidatos a partir de inferencias. Las relaciones son descritas en lenguajes RDF para su fácil manejo con razonadores y sencilla integración a la Web de datos.

El trabajo propuesto ofrece una perspectiva completamente semántica y carece de verificaciones estructurales entre los tipos de datos requeridos en los mensajes de entrada y salida de las operaciones analizadas. Es gracias a esto que la técnica propuesta ofrece buenos resultados en la selección de SW con posibilidades de composición para la resolución de un objetivo. Sin embargo, los resultados arrojados no ofrecen certeza en la composición, ya que el proceso nunca realiza comparaciones de interfaces, lo cual impide conocer con exactitud si dos operaciones pueden ser serializadas en un flujo de trabajo para un servicio compuesto.

Zakaira Maamar et al. [4], Identifican las desventajas de poseer servicios de descubrimiento basados en repositorios de definiciones clasificadas y proponen el uso de técnicas basadas en redes sociales para el pronto descubrimiento de servicios. Para hacer uso del concepto de redes sociales en la recomendación de SW, los autores proponen una técnica para mantener índices de recomendación entre los propios SW, donde cada servicio es capaz de recomendar servicios web cuyas características sean iguales o parecidas a las funcionalidades brindadas por el mismo.

Los autores identifican seis diferentes pasos para establecer un servicio de recomendación entre SW basados en las similitudes semánticas que estos posean, dichos pasos son: Identificación de componentes sociales, análisis de similitud entre SW, administración social de los SW, evaluación de nodos, navegación sobre la red social y evaluación de nodos externos. En general los pasos definidos, construyen un grafo donde los nodos son las especificaciones de los SW y los arcos las relaciones de similitud entre ellos, adicionalmente, los pasos mantienen actualizado el grafo de recomendación a través de un análisis de similitud entre los nuevos SW que hayan sido incorporados al servicio de descubrimiento.

Finalmente los autores realizaron pruebas comparativas entre el servicio de descubrimiento UDDI y el servicio basado en redes sociales propuesto, cuyos resultados mostraron ventajas contundentes en las recomendaciones de SW realizadas por el método de redes sociales sobre los resultados otorgados por UDDI.

Capítulo 2

Marco Teórico

La comparación de dos entidades consiste en establecer una relación de semejanza entre dos partes, descubriendo sus relaciones y/o estimando sus diferencias y semejanzas [16]. Existen diversas aplicaciones que requieren de la comparación de elementos para la toma de decisiones [17]. La correlación de señales hecha por un radar para localizar un blanco o la comparación de estructuras proteicas para su correcta clasificación y estudio, son dos ejemplos aislados de aplicaciones cuyo fundamento radica en las comparaciones que estos realizan.

La comparación de servicios Web se ha realizado con la finalidad de proporcionar métodos capaces de clasificar y permitir búsquedas específicas en repositorios de servicios Web. Comparar servicios Web no solo permite clasificarlos, sino también obtener listas de servicios posiblemente reemplazables o completamente intercambiables entre sí, lo que permite un mayor dinamismo en la selección de uno o varios servicios Web para construir aplicaciones o servicios compuestos que hagan uso de ellos.

Un servicio Web no es capaz de describirse a sí mismo, es por ello que requiere de un archivo de definición WSDL para especificar todos los elementos que lo componen y permiten su invocación. Un archivo WSDL es un tipo de documento XML, por lo tanto la descripción de un servicio Web es textual y a la vez estructurada. Las comparaciones entre servicios Web se basan en la extracción estructural y en la interpretación de las descripciones textuales contenidas en los archivos WSDL. Existen tres variantes en las comparaciones de servicios Web, las cuales radican en: realizar comparaciones semánticas entre las operaciones de un servicio, realizar comparaciones entre las estructuras XML implícitas en las definiciones de las operaciones de un servicio o realizar una comparación semántica y estructural en combinación.

Los diferentes métodos de comparación para servicios Web poseen sus propias ventajas. Las comparaciones semánticas determinan una relación entre operaciones cuando sus descripciones textuales son similares, es decir, una comparación semántica establece grados de similitud con base en el obje-

tivo y la funcionalidad que cada operación describe. Cada operación de un servicio Web se encuentra definida por diferentes tipos de datos, ya sean simples o complejos, los cuales se encuentran representados en una estructura de árbol provista por el propio documento XML. Con la comparación de estructuras de árbol es posible determinar si dos operaciones de servicios Web poseen la misma estructura de invocación, por lo tanto las comparaciones estructurales determinan qué tan remplazables son dos operaciones. Por otro lado, las comparaciones semántico-estructurales proporcionan las ventajas individuales de cada técnica de comparación, sin embargo, al estar estrechamente integradas, no permiten la mejora individual para sus métodos de comparación, complicando sus posibles optimizaciones o remplazos.

2.1. XML

XML (por sus siglas en ingles *eXtensible Markup Language*) es un conjunto de reglas para definir etiquetas semánticas que dividen e identifican a un documento en partes a través de un metalenguaje de etiquetado [18]. XML define la sintaxis de marcas o etiquetas para clasificar, jerarquizar y ordenar información de algún dominio específico, facilitando su entendimiento e intercambio.

Otros lenguajes de marcas como TeX, troff o HTML definen una cantidad fija de etiquetas que describen a un conjunto finito de elementos. Por lo cual, no son capaces de definir contenidos de manera específica para diferentes contextos.

El metalenguaje de XML permite organizar un documento mediante etiquetas acordes a un contexto específico. Por ejemplo, si se quiere almacenar la información referente al árbol genealógico de una familia, se necesitará definir nombres, edades, fechas, cumpleaños, adopciones, bodas, etc. Para lo cual XML permitirá definir una etiqueta específica para cada una de estas categorías, sin la necesidad de forzar la inserción de información a una etiqueta de contexto diferente, provista por un lenguaje de marcas fijas. Por lo anterior, XML permite definir tantas etiquetas de contextos diferentes como sean requeridas para ser desplegadas mediante intérpretes de XML.

XML especifica reglas de sintaxis para distinguir entre el contenido de las marcas XML y los atributos relacionados con dicho contenido [18], estableciendo patrones que las etiquetas deben seguir. Es decir, XML interpreta como elementos a las oraciones que comiencen con el signo “<” y terminen con el signo “>” e interpreta como contenido a todas las oraciones contenidas entre elementos de apertura y clausura. Un elemento de apertura se define por los signos “<”, “>” y un nombre de identificación entre ellos. Así mismo, un elemento de clausura se define exactamente igual que su correspondiente etiqueta de apertura, colocando adicionalmente el símbolo “/” entre al signo “<” y el nombre de la etiqueta. Además, XML especifica reglas de estructuración que permiten el anidamiento de etiquetas, propiciando una jerarquización inherente en la información de un documento.

Las aplicaciones de XML se clasifican principalmente en dos tipos:

1. Aplicaciones de documento que manejan información que va dirigida principalmente a usuarios que realizarán consultas de manera manual.

2. Aplicaciones de datos que manejan información que va dirigida principalmente a aplicaciones de software que explotarán la información de manera automática.

La principal ventaja de XML en aplicaciones para documentos es que XML se concentra en la estructura del documento y su representación, haciéndolo independiente del formato de entrega o presentación, el cual puede ser HTML, XHTML, PostScript, PDF, etc. Esto ayuda al usuario a acceder a la información de la manera en que mejor le convenga. Esta característica es por la cual XML es muy utilizado en páginas Web, ya que permite editar y mantener documentos XML y publicarlos automáticamente en Internet.

En las aplicaciones de datos, XML facilita el acceso a documentos mediante varias herramientas de software de procesamiento automático, lo que permite compartir información con sistemas de bases de datos de manera sencilla. XML permite ingresar información a una base de datos, o al contrario, obtener información de una base de datos para generar un documento XML, lo cual es muy útil para intercambiar información entre organizaciones.

Al publicar información en la Web utilizando XML, se pueden utilizar herramientas para visitar un sitio de manera automática, extraer información específica y procesarla de acuerdo a una tarea delimitada con anterioridad. Todo esto es posible gracias a que se conocen las etiquetas de los documentos de interés y permiten ubicar la información requerida rápidamente.

Cuando se utiliza XML, se pueden aprovechar varios estándares desarrollados por la W3C (*World Wide Web Consortium*), estos se conocen con el nombre de estándares acompañantes de XML, algunos de ellos son los siguientes:

- DOM y SAX: se tratan de APIs que se utilizan para acceder a documentos XML; éstas permiten a las aplicaciones leer documentos XML sin tener que preocuparse por la sintaxis. DOM (Modelo de Objetos de Documento) es la representación estándar de un documento en memoria, en la cual se representa un documento como un árbol de nodos, mientras que SAX (API Simple para XML) crear un árbol de un documento XML, lee el archivo y ejecuta oyentes registrados que perciben cuándo ocurren ciertos eventos del análisis.

- Hojas de estilo: especifican cómo deben presentarse en pantalla, en papel o en un editor los documentos XML. Es soportado por dos lenguajes XSL (Lenguaje de Hojas de Estilo Extensible) y CSS (Hojas de Estilo en Cascada).

La manera adecuada de desarrollar documentos XML es que éstos se encuentren bien formados y sean válidos. Se dice que un documento bien formado es aquel que obedece un orden jerárquico para todos y cada uno de sus elementos, mientras que un documento es válido, si primero está bien formado y, después, cada uno de sus elementos obedece una cierta secuencia lógica descrita en un archivo de definición [18].

Para la definición de la estructura de documentos XML y las etiquetas a emplear en su elaboración, se pueden utilizar archivos llamados DTD (Definición de Tipos de Documentos) o de esquema (XML Schema). Un archivo DTD permite definir y asignar un nombre a los elementos que se pueden utilizar en el documento, el orden en el cual podrán aparecer dichos elementos y los atributos de los elementos que podrán utilizarse. Los elementos de un DTD indican cuáles serán las etiquetas que se podrán usar dentro del documento XML y en qué orden.

XML es un lenguaje de propósito general para estructurar información. Por lo cual, deja de lado características no esenciales que pueden ser provistas por complementos de XML. Descripciones para brindar estilo a documentos etiquetados o lenguajes enriquecidos para etiquetados de mayor precisión, son características delegadas a complementos de XML. Complementos como XSL que describe la manera en la que un documento XML deberá ser mostrado, XLink que incorpora vocabularios hipertextuales o XML Schema que permite definir tipos de datos para etiquetados de XML, son algunos refinamientos existentes para XML.

XML Schema

XML Schema facilita la descripción y restricción de datos sobre la información contenida en un documento XML. Los documentos XML carecen de restricciones debido a la flexibilidad que tienen para estructurar información con etiquetas adaptables a cualquier contexto, por lo cual la validación de la información del documento es complicado.

A diferencia de XML, XML Schema define la estructura de sus documentos en una versión mejorada de DTD, llamada espacio de nombres. Los espacios de nombres aportan un gran número de tipos de datos predefinidos para hacer restricciones sobre el etiquetado de cada documento [19]. Además, permiten crear tipos de datos complejos para adaptarlos a las necesidades de una aplicación específica. La especificación de tipos de datos en documentos XML aumenta la compatibilidad de intercambio de información con aplicaciones de procesado de datos, debido a la estandarización de los mismos.

Un documento sobre información bancaria escrito en XML puro puede tener una etiqueta como la siguiente: <BALANCE_CTA>$100,000.99</BALANCE_CTA> y únicamente expresar que el contenido de la etiqueta es una cadena de caracteres, impidiendo su uso inmediato en aplicaciones de cálculos financieros que deberán transformar el contenido en un tipo de dato flotante o doble para

realizar operaciones sobre la cantidad. Por otro lado, XML Schema permite establecer restricciones de tipos sobre el etiquetado de un documento, haciendo compatible la información de un documento con los tipos de datos requeridos por una aplicación, permitiendo así su uso inmediato.

Finalmente, un archivo Schema permite escribir esquemas detallados para los documentos XML, utilizando una sintaxis estándar de XML. De modo que, permite utilizar tipos de datos primitivos o crear nuevos tipos de datos, limitando los errores que pudieran presentarse al momento de generar etiquetados, mediante la validación de la información permitida por cada etiqueta definida en el espacio de nombres. Los elementos del espacio de nombres de un documento XML Schema, al igual que en los DTD, indican cuáles serán las etiquetas que se podrán usar dentro del documento XML.

2.2. RDF

RDF (*Resource Description Framework*) es un método para descomponer y representar conocimiento en piezas pequeñas, con algunas reglas acerca de la semántica o significado de las mismas. Es un método simple que permite expresar cualquier hecho en una pieza básica de conocimiento, manteniendo una estructura relacional entre los elementos de conocimiento para que diferentes aplicaciones puedan hacer uso de él.

El método provisto por RDF requiere identificar las relaciones simples entre objetos, donde una relación simple está descrita por tres elementos, un sujeto, un predicado y un objeto y se conocen como tripletas [20]. RDF representa a las tripletas como grafos dirigidos, donde cada arista del grafo representa un hecho o relación entre dos objetos o vértices (figura 2.1).

Figura 2.1: Representación de una tripleta en RDF

En cierta forma, RDF puede ser comparado con XML, ya que también está diseñado para ser simple y aplicable a cualquier tipo de datos. Sin embargo, XML por sí solo no está relacionado con la estructuración de conocimiento. Lo que implica que RDF ha sido concebido particularmente para brindar significado a la información.

RDF permite representar conocimiento distribuido, es decir, las aplicaciones RDF pueden juntar archivos RDF publicados por diferentes autores en Internet y construir nuevo conocimiento a partir de ellos, propiciando la reutilización de información [21]. Enlazando documentos que utilicen vocabularios comunes o permitiendo que cualquier documento use cualquier vocabulario, Brindando una gran flexibilidad al momento de expresar hechos sobre una amplia gama de objetos, basándose en información proveniente de una amplia variedad de fuentes.

Toda información contenida en un documento RDF tiene un significado. Puede ser una referencia a un objeto en el mundo real, como una persona o un lugar, o puede ser un concepto abstracto, como la relación de amistad entre dos personas.

RDF permite establecer una representación de conocimiento, empezando por una lógica de predicados que eventualmente puede convertirse en una red semántica; de igual manera se puede utilizar en la descripción de esquemas para bases de datos y otros modelos de datos.

Principalmente, el esquema RDF se ha utilizado para la recopilación de información de bibliotecas digitales y la definición de los metadatos que puedan definirlos. Mediante RDF se ha definido una serie de vocabularios o compendios de conocimiento que permiten un mejor procesamiento e inferencia de información; estos vocabularios pueden utilizarse de manera independiente para enriquecer a otros compendios de información.

Entre las aplicaciones de RDF, se pueden mencionar las siguientes:

- Recuperación de información mediante motores de búsqueda.
- Intercambio de información en forma de conocimiento generado mediante software inteligente.
- Representación de documentos lógicos.
- Elaboración de catálogos para descripción de contenidos.
- Definición de los derechos de autor de las páginas Web.

2.3. SOAP

SOAP (*Simple Object Access Protocol*) es un protocolo para el intercambio de información estructurada entre sistemas de información en un entorno distribuido [11]. Utiliza XML para describir el formato de los mensajes de intercambio y su uso habitual es en servicios Web. SOAP se apoya en dos principios que son la simplicidad y la extensibilidad. Para conseguirlos, SOAP ignora ciertos aspectos relativos a los sistemas distribuidos que pueden ser extendidos cuando sea necesario por otras especificaciones. La especificación de SOAP que describe la infraestructura de mensajes consta de tres partes:

1. El modelo de procesos SOAP, que define el mecanismo necesario para el procesamiento de mensajes.
2. El modelo de extensión de SOAP, que se ocupa de las características y módulos que pueden ser extensibles.

3. El protocolo de enlace con el marco de trabajo, que establece la reglas para asociar a SOAP con un protocolo de bajo nivel que permita el intercambio de mensajes entre nodos SOAP.

El modelo de comunicación de SOAP está definido como un sistema distribuido, en el que intervienen diferentes nodos [22]. En un escenario básico, los nodos SOAP se comunican asumiendo roles de emisor y receptor. No obstante, SOAP permite interacciones más complejas mediante diferentes tipos de nodos que asumen diversas condiciones en el envío de un mensaje.

Uno de los objetivos de SOAP es proporcionar los medios para llevar a cabo llamadas a procedimientos remotos [11], usando para ello las posibilidades de extensión de XML. SOAP contempla la gestión de las situaciones en las que se puedan producir fallos durante el intercambio de mensajes. Además, distingue entre las condiciones que provocan el fallo y la capacidad para reportar el error al origen de la situación de fallo o a cualquier otro nodo SOAP. En cuanto a la asociación de SOAP con un protocolo de más bajo nivel para el intercambio de mensajes, el protocolo elegido habitualmente es HTTP.

2.4. Servicios Web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible definición sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web [9]. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio, llamando a estos procedimientos a través de la Web.

¿Para qué sirven? Estos servicios proporcionan mecanismos de comunicación estándar entre diferentes aplicaciones que interactúan entre sí, para presentar información dinámica al usuario. Proporcionan interoperabilidad y extensibilidad entre aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas. Ofrecen servicios que resuelvan operaciones de propósito específico y que puedan ser invocadas por cualquier usuario que requiera de ellas [22].

Servicios Web semánticos

Un servicio Web semántico es un servicio Web enriquecido con metainformación para facilitar su búsqueda y composición de forma automática. Los servicios Web semánticos suponen la integración de la Web semántica y los servicios Web. Un servicio Web semántico extiende el concepto de servicio Web, dotándolo de aspectos semánticos que permiten su uso automático en sistemas de información con acceso a la Web [22].

Las tecnologías empleadas en los servicios Web semánticos emplean descripciones formales y razonamiento automático para ofrecer posibilidades descriptivas en la definición de interfaces de un servicio Web.

2.5. WSDL

WSDL [23] es un lenguaje basado en XML para describir servicios Web y es una recomendación del World Wide Web Consortium [22]. Los documentos WSDL describen un servicio Web especificando los métodos que éste ofrece. Estos métodos se describen de forma abstracta. Para el uso de los servicios hay que emplear algún protocolo de comunicaciones, el cual no está impuesto por WSDL. El uso de los servicios se materializa mediante el intercambio de mensajes que contienen información de tipo procedimental. El formato de estos mensajes tampoco viene determinado por WSDL. Esta separación entre la especificación de los servicios y la implementación de éstos es uno de los aspectos más relevantes de WSDL.

Estructura de documentos WSDL

Los elementos principales de WSDL para la definición de servicios son [24]:

Listado 2.1: Estructura básica de un documento WSDL

```
<?xml version="1.0" enconding="UTF-8"?>
    <wsdl:definitions>
        <wsdl:types>
                ...
        </wsdl:types>
        <wsdl:message>
                <part name="parametro" type="xsd:string"/>
        </wsdl:message>
        <wsdl:portType>
                <wsdl:operation name="funcion">
                        ...
                </wsdl:operation>
        </wsdl:portType>
        <wsdl:binding>
                ...
        </wsdl:binding>
        <wsdl:service name="HelloWorldService">
                ...
        </wsdl:service>
    </wsdl:definitions>
```

- Tipos: permite definir tipos empleando algún sistema de tipos, como XML Schema (XSD).
- Mensaje: una descripción de los datos intercambiados cuando se hace uso de los servicios.

- Operación: una funcionalidad ofrecida por el servicio.
- Tipo de puerto (*Port Type*): una conjunto de operaciones abstractas ofrecidas por uno o más puntos finales (*endpoint*).
- *Binding*: un protocolo de comunicaciones y un formato de mensajes concretos utilizados por un tipo de puerto.

Un documento WSDL posee la siguiente estructura, donde pueden ser fácilmente apreciados los elementos descritos con anterioridad [24]:

WSDL es el lenguaje habitual para describir servicios Web. Su gran acierto radica permitir una descripción abstracta de servicios, sin ligarlos a una implementación concreta, lo que ofrece las siguientes ventajas:

- Permite el uso de cualquier implementación.
- Es posible reutilizar la definición de estos servicios con diferentes implementaciones y escenarios.
- Facilita la interoperabilidad cuando existen implementaciones diferentes de los mismos servicios.

A pesar de las ventajas anteriores, desde el punto de vista semántico, WSDL carece de recursos para describir aquellos aspectos de los servicios Web que vayan más allá de su interfaz o su implementación [25]. Los aspectos semánticos que describen a un servicio y a las condiciones que se tienen que dar para que un servicio pueda ser prestado, quedan fuera del alcance de WSDL. El hecho de dejar fuera los aspectos semánticos de una especificación de servicios Web hace que WSDL sea más sencillo de entender y usar, permitiendo una rápida adopción. Por otro lado, hay que decir que para describir semánticamente a un servicio Web se requieren de otros documentos, los cuales suelen ser textuales y no estar estandarizados, lo que provoca que en muchos casos estos documentos sean incompletos y/o ambiguos [26].

2.6. Descubrimiento de servicios Web

El descubrimiento de servicios Web es un proceso que consiste en localizar o descubrir, uno o varios documentos relacionados con la descripción de un servicio Web determinado [24], realizando comparaciones entre un conjunto de especificaciones de servicios Web y un patrón de búsqueda, ya sea a través de un archivo WSDL o una simple descripción semántica. A través del proceso de descubrimiento, los usuarios obtienen la localización de los archivos WSDL de los servicios Web que poseen un alto grado de similitud con el patrón de búsqueda que proporcionaron.

Los sitios que implementan servicios Web no necesitan ser compatibles con aplicaciones de descubrimiento de servicios, ya que ellos solo poseen la implementación del servicio y no su definición.

Gracias a que los servicios Web por sí solos no contienen una definición de ellos mismos y requieren de una especificación WSDL que los defina, es posible la creación de repositorios de servicios Web que se encuentren almacenados en lugares físicos independientes a los servidores donde se encuentran las implementaciones de los servicios Web que describen. Haciendo que las aplicaciones para el descubrimiento de servicios Web se ocupen de organizar, clasificar y buscar servicios Web en repositorio, y devolviendo como resultado de una búsqueda los archivos de definición que han cumplido con el criterio de búsqueda.

2.7. Estadística

Cuando se tiene un conjunto de mediciones representativas de un evento, fenómeno y/o experimento, es necesario analizarlo para otorgarle un significado. El análisis los datos recabados permite caracterizar y dotar de interpretación al fenómeno que originó la recolección de mediciones. La representación, la medición y la estimación de información, a partir de una colección de datos relacionados, son partes fundamentales de la estadística que estructuran información y habilitan la toma de decisiones.

La estadística se divide en dos categorías con perspectivas diferentes en el análisis de datos. Por un lado, se encarga de representar y medir los datos asociados a una colección y por otro se encarga de brindar interpretaciones lógicas al comportamiento que una colección de datos puede tener, clasificándose en estadística descriptiva y estadística inferencial.

La estadística descriptiva está formada por procedimientos empleados para resumir y describir las características importantes de un conjunto de mediciones [27].

La estadística inferencial está formada por procedimientos empleados para hacer inferencias acerca de las características poblacionales, a partir de información contenida en una muestra sacada de una población [27].

2.7.1. Datos

Para realizar un experimento o medición, es necesario comprender el concepto de dato. Los datos son representaciones simbólicas de atributos o variables que contienen una pequeña pieza de información [28]. En conjunto, los datos modelan y describen a las variables o atributos de estudio.

Una variable es una característica que cambia o varía con el tiempo y/o para diferentes personas u objetos bajo consideración [27].

Si se genera una medición para una variable experimental, se obtiene una colección de datos, la

cual es llamada la población de interés de la variable. Cualquier conjunto más pequeño de mediciones sobre la misma variable es llamado muestra.

Podemos definir a una población como el conjunto de mediciones de interés para el investigador y a una muestra como un subconjunto de mediciones seleccionado de la población de interés.

La estadística analiza los datos asociados a una o más variables, destacando el patrón de comportamiento existente entre ellas.

Resultan datos univariados cuando se mide una sola variable en un experimento. De igual manera, resultan datos bivariados cuando se miden dos variables en un experimento. Y son llamados datos multivariados cuando se miden más de dos variables estadísticas [27].

2.7.2. Medidas de centro y variabilidad

Los conjuntos de datos pueden ser graficados para mostrar su distribución respecto a una unidad medible. La figura 2.2 muestra la variación de las mediciones de una variable X cualquiera a lo largo de un intervalo de tiempo, lo cual tiene una distribución gaussiana de media $\overline{X}$ y desviación estándar s.

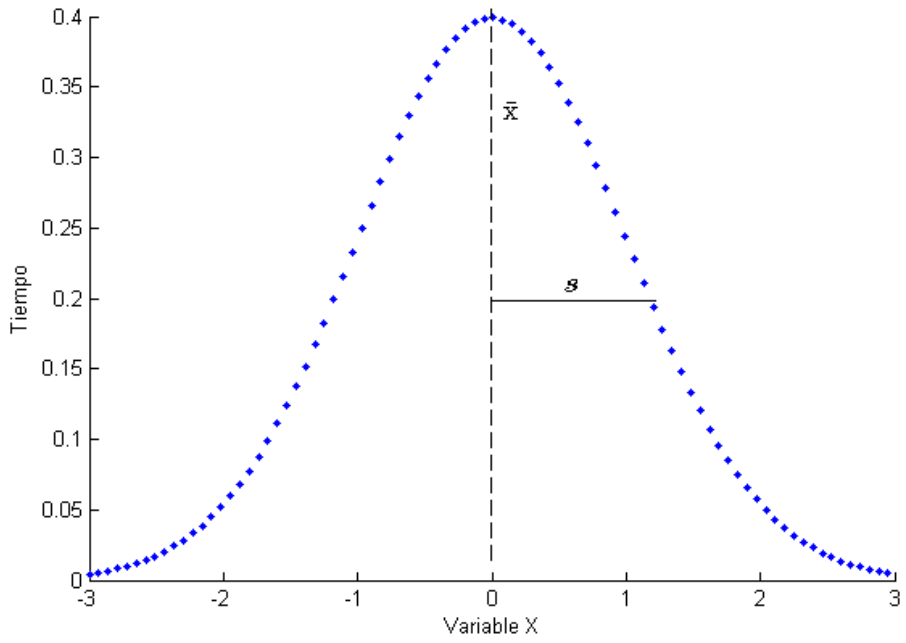


Figura 2.2: Distribución de una variable X

La variabilidad o dispersión es una característica muy importante de los datos, ya que otorga una visión de las distancias existentes entre los datos de una muestra y permite observar gráficamente el comportamiento de una variable.

La estadística ofrece medidas de variabilidad o dispersión para caracterizar los datos de una muestra y ofrece medidas de centro para ubicar el punto medio de una distribución de datos.

Las medidas de centro más importantes son la media y la mediana, las cuales pretenden descubrir el centro de una distribución de datos.

La **media** representa el promedio de los valores de una muestra y su fórmula es la siguiente:

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.1}$$

La **mediana** es el elemento que ocupa la posición central entre los valores de una muestra ordenada, lo cual es fácilmente detectable en muestras de tamaño impar, sin embargo en muestras de tamaño par es necesario realizar el promedio entre los dos valores centrales de la muestra.

Ambas medidas estiman el centro de una distribución con diferente precisión. Debido a que la mediana depende en gran medida de las magnitudes que posean los valores de una muestra; la mediana puede calcular inadecuadamente el centro de una distribución, cuando exista una dispersión alta. Por otro lado, la media proporciona el promedio de la muestra, otorgando mayor precisión en la detección del centro de una distribución.

Las medidas de dispersión más importantes son el rango, la desviación, la varianza y la desviación estándar, las cuales determinan la relación de dispersión existente entre los datos de una muestra.

El **rango** R de un conjunto de n mediciones se define como la diferencia entre la medición más grande y la más pequeña.

$$R = x_{max} - x_{min} \tag{2.2}$$

Para medir la variabilidad de un conjunto de mediciones es necesario tener un punto de referencia en la distribución de datos. La definición de un punto de referencia permite establecer qué tan dispersas se encuentran las mediciones de una muestra. Se utiliza la media como el punto central de la distribución y sobre la cual es posible estimar la dispersión de los datos de una muestra.

Las distancias horizontales entre cada punto (medición) y la media $\overline{x}$ ayudan a medir la variabilidad de los datos (figura 2.3). Si las distancias son grandes, los datos son más dispersos o variables que si las distancias son pequeñas. La distancia que posee cada punto de una distribución de datos hacia la media de la misma $(x_i - \overline{x})$, es llamada **desviación**.

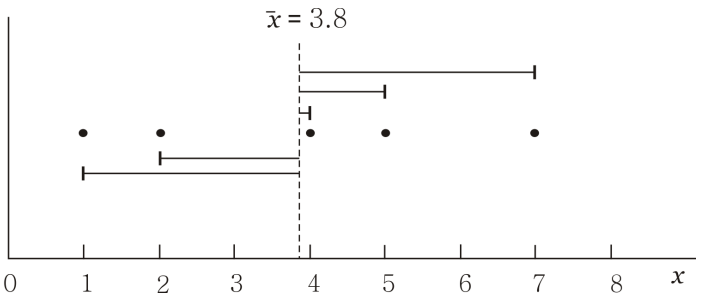


Figura 2.3: Desviaciones de una muestra Y

Una forma para combinar las desviaciones de una muestra en una única medida numérica es promediarlas. Desafortunadamente, el promedio no funcionará porque algunas de las desviaciones son positivas, algunas negativas y la suma siempre es cero. Para superar la dificultad causada por los signos de las desviaciones es necesario trabajar con la suma de sus cuadrados. De la suma de desviaciones cuadradas, se calcula una sola medida llamada **varianza**. *La varianza será relativamente grande para datos muy variables y relativamente pequeña para datos menos variables.*

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1} \tag{2.3}$$

$$s^2 = \frac{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}{n-1} \tag{2.4}$$

Varianza computacional

La varianza se mide en términos del cuadrado de las unidades originales de medición. Si las mediciones originales son pulgadas, la varianza se expresaría en pulgadas cuadradas. Tomando la raíz cuadrada de la varianza, obtenemos la **desviación estándar**, que regresa la medida de variabilidad a las unidades originales de medición. Por lo tanto, la desviación estándar de un conjunto de mediciones es igual a la raíz positiva de la varianza.

$$s = \sqrt{s^2} \tag{2.5}$$

2.7.3. Gráficas de dispersión para datos bivariados

Es muy frecuente que investigadores se interesen en más de una variable que pueda medir su investigación. Cuando dos variables se miden en un solo experimento, los datos resultantes se denominan datos bivariados. Los métodos para graficar y analizar datos bivariados permiten estudiar a

las dos variables juntas.

Cuando las dos variables que han de presentarse en una sola gráfica son cuantitativas, una de ellas se grafica a lo largo del eje horizontal y la otra a lo largo del eje vertical. De modo que la gráfica posea la forma de una gráfica con puntos (x, y). Cada para de valores de datos se grafica como punto en una grafica de dos dimensiones llamada **grafica de dispersión**.

A través de una gráfica de dispersión es posible describir la relación entre dos variables x y y, observando los patrones que se muestran en la gráfica, interpretando a través de la localización de los puntos la similitud existente entre los valores de cada una de las variables.

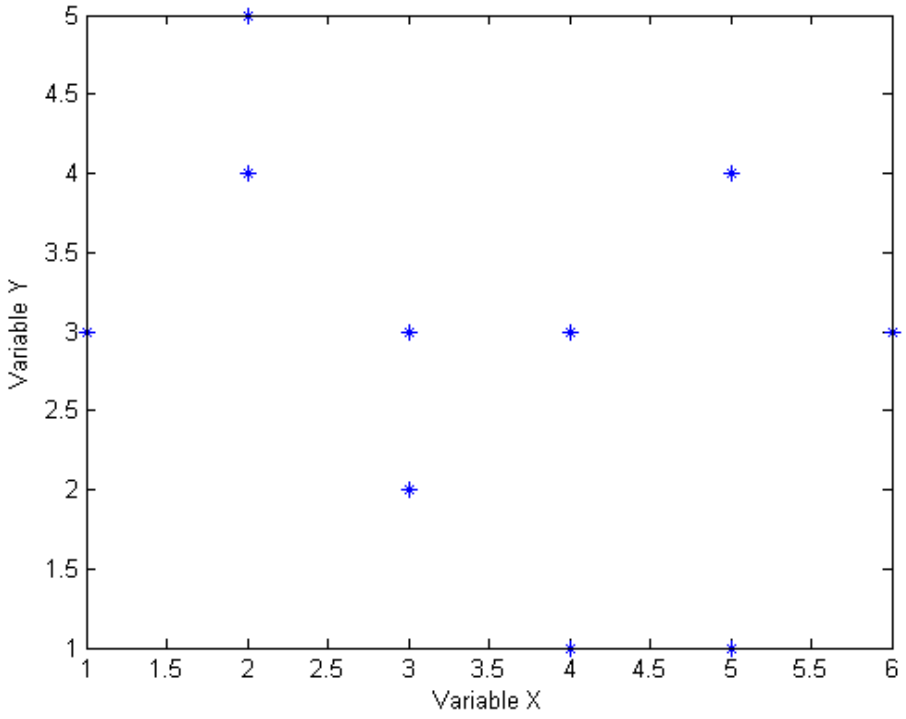


Figura 2.4: Grafica de dispersión para las variables X y Y.

La figura 2.4 muestra una gráfica de dispersión para las variables $X = \{4, 5, 3, 4, 5, 3, 6, 1, 2, 2\}$ y $Y = \{3, 4, 3, 1, 1, 2, 3, 3, 4, 5\}$.

2.7.4. Correlación

Una gráfica de dispersión exhibe un modelo entre dos variables estadísticas, dibujando modelos gaussianos, lineales cuadráticos, etc. lo que describe la relación existente entre cada par de valores. Para cada par de valores se podría describir individualmente cada variable x y y, usando medidas descriptivas como son las medias $\overline{x}$ y $\overline{y}$ o las desviaciones estándar s_x y s_y. No obstante estas medidas

no describen la relación entre x y y.

Una medida que sirve a este propósito se denomina coeficiente de correlación, la cual proporciona un valor en el rango de $[-1, 1]$ para establecer el nivel de variación conjunta existente en entre dos variables estadísticas, a través de la normalización de la covarianza. El coeficiente de correlación se denota por r y se define como:

$$r = \frac{s_{xy}}{s_x s_y} \tag{2.6}$$

Donde los valores s_x y s_y son las desviaciones estándar para las variables x y y, respectivamente. El valor s_{xy} se denomina **covarianza** entre x y y, y mide el grado de variación conjunta entre dos variables estadísticas. La covarianza se define como:

$$s_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{n-1} \tag{2.7}$$

La covarianza implica la multiplicación de las desviaciones de los puntos con la misma posición i dentro de las variables estadísticas x y y, de lo cual, podemos destacar un hecho evidente en el cálculo de la desviación. Cuando el valor z_i de una desviación $(z_i - \overline{z})$ es menor a la media $\overline{z}$, el valor de la desviación tendrá signo negativo y en el caso contrario la desviación será positiva. Por lo anterior, una gráfica de dispersión bivariada puede ser dividida en cuatro áreas (figura 2.5) que otorgaran el signo a cada multiplicación $(x_i - \overline{x})(y_i - \overline{y})$ en una covarianza.

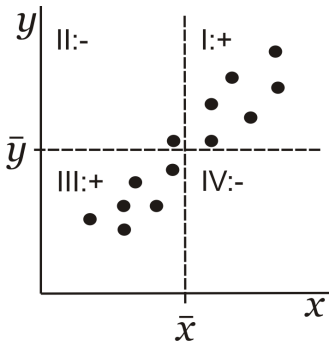


Figura 2.5: Modelo Positivo

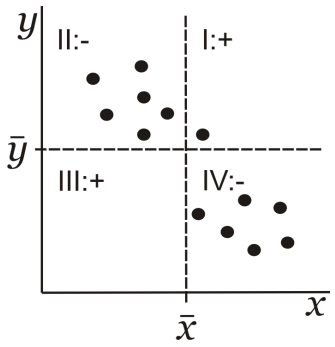


Figura 2.6: Modelo Negativo

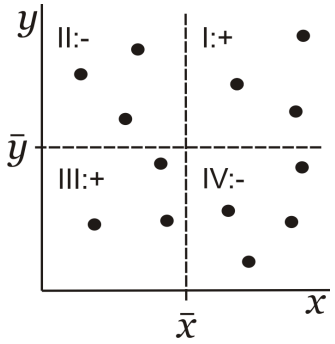


Figura 2.7: Sin modelo

Finalmente, la covarianza proporciona una medida de relación entre dos variables estadísticas, las cuales grafican un modelo de dispersión especifico que muestra qué tan parecidas son dos variables.

La similitud entre dos variables estadísticas se puede apreciar en una gráfica de dispersión cuando los puntos situados en la grafía tienden a formar diagonales con pendientes positivas (figura 2.5) o negativas (figura 2.6), implicando que los valores de cada variable son cercanos o iguales. Por otro lado, cuando no existe un patrón en los puntos de la gráfica (figura 2.7), se dice que la gráfica carece de un modelo, por lo cual la relación entre dos variables es mínima o nula.

Capítulo 3

Análisis de similitud entre estructuras de árbol N-arias provenientes de operaciones de servicios Web.

El presente trabajo aborda la comparación estructural entre operaciones de servicios Web, separando la comparación semántica de la estructural y a su vez permitiendo una integración modular entre ellas, con la finalidad de brindar flexibilidad a las aplicaciones de comparación de servicios Web en la selección de algoritmos de comparación.

Para lograr una separación modular entre comparaciones semánticas y estructurales así como su fácil adopción, proponemos el uso de estructuras de árbol que representen operaciones de SW mediante objetos Java y archivos de connotación semántica RDF para la descripción de operaciones, cuyos archivos son ampliamente utilizados en la Web de datos y permiten hacer inferencia lógica sobre los datos que representan. Los archivos RDF se componen de una red estructurada de tripletas o relaciones entre datos, las cuales brindan significado semántico a la información que poseen.

El método descrito en este capítulo para construir árboles de representación a partir de la descripción de cada servicio Web, analiza documentos WSDL en búsqueda de las operaciones que componen a los servicios, extrae sus estructuras y las representa en árboles que pueden ser comparables por el sistema. Cada árbol es serializado en un archivo RDF y a su vez cada serialización RDF es recuperable en su representación de árbol comparable.

Nuestra propuesta de comparación estructural entre operaciones de servicios Web, recupera las

estructuras de árbol de las operaciones contenidas en las especificaciones de los SW a comparar. Para después calcular árboles complementarios que homogeneizaran las estructuras a comparar por cada operación, normalizandolas para su posterior comparación.

Definimos a un súper árbol como un árbol que contiene a las dos estructuras de la comparación, y definimos a un árbol complementario como el árbol comparable recuperado de la serialización RDF aumentado con nodos vacíos para igualar la estructura del súper árbol de la comparación.

La propuesta de comparación realiza una correlación entre los vectores característicos de los árboles complementarios de cada operación, los cuales se obtienen de un recorrido en pre orden de los árboles complementarios previamente pesados, calculándose finalmente un nivel de correlación o similitud entre dos operaciones.

3.1. Extracción de operaciones de un servicio Web

Un servicio Web puede entenderse como un conjunto de aplicaciones o de tecnologías con capacidad para inter-operar en la Web [1], es decir, son colecciones de métodos en espera constante de ejecución a través de la Web, tal como si se tratase de un método remoto. Al igual que un método remoto una operación de un servicio Web posee una interfaz de invocación y es requisito conocer dicha interfaz al momento de solicitar su ejecución.

Para conocer las interfaces propias de las operaciones que un servicio Web ofrece, es necesario consultar el archivo descriptivo WSDL del servicio, el cual representa una especie de contrato entre el proveedor de un servicio y un solicitante. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes, para la invocación de cada operación de un servicio.

Debido a que es necesario explorar la definición WSDL de un SW para conocer su estructura, extraer la información de las operaciones de WS a partir de sus archivos de definición es un paso obligado para todas las aplicaciones de organización y clasificación de SW. Por lo anterior, proponemos un módulo estándar para la extracción y representación de operaciones, que permita homogeneizar el formato de las operaciones entre las aplicaciones de clasificación y organización de SW, y permita hacer compatibles sus algoritmos de comparación.

3.1.1. Extracción de operaciones

Para realizar una comparación entre operaciones de servicios Web, es necesario buscar las interfaces de las operaciones involucradas en una comparación. Además, es necesario extraer la estructura de tipos necesaria para la invocación y recepción de resultados de las operaciones del servicio.

[1] Guía Breve de Servicios Web, W3C, http://w3c.es/Divulgacion/GuiasBreves/ServiciosWeb
Julio 2014

Un documento WSDL mantiene la especificación de un servicio Web perfectamente estructurada, propiciando que su información sea legible e inteligible no sólo para los ordenadores, sino también para las personas. WSDL es un tipo de documento XML, por lo cual hereda las ventajas de organización y lectura de XML, facilitando su exploración a través de un modelo de objetos como DOM. El modelo de objetos para la representación de documentos DOM, extrae el árbol representativo de un documento de marcas tal como WSDL. Debido a que la estructura natural de un documento de marcas es la de un árbol no binario, decidimos mantener dicha estructura en la representación de operaciones.

Para comparar operaciones de servicios Web, construimos un árbol por cada operación de SW, cuyos nodos son los elementos estructurales recuperados por la extracción de interfaces de los archivos WSDL. Reduciendo el problema de la comparación estructural a una comparación de árboles no binarios.

Las comparaciones entre operaciones de SW, se basan en obtener la información etiquetada referente a las interfaces de las operaciones de cada servicio, por lo cual es necesario identificar las etiquetas (Tabla 3.1) que hagan referencia a dichas operaciones.

Etiqueta	**Función**
<wsdl:PortType>	Define a un servicio Web y a las operaciones de las que dispone.
<wsdl:operation>	Especifica cada operación de un servicio Web.
<wsdl:input>	Define el mensaje de invocación de una operación.
<wsdl:output>	Define el mensaje de respuesta de una operación.
<wsdl:message>	Define la interfaz del mensaje de invocación para el método remoto de la operación.
<wsdl:types>	Delimita a un documento XML Schema para la definición de tipos de dato utilizados en las operaciones definidas en el archivo WSDL.

Cuadro 3.1: Etiquetas importantes en la definición de operaciones de un servicio Web.

Nuestra propuesta analiza únicamente las etiquetas de la tabla 3.1 y extrae la información que éstas contienen para convertirlas en nodos de un árbol. Las etiquetas de la tabla 3.1 proporcionaran la estructura base para la definición de operaciones de SW. Sin embargo, solo definen operaciones hasta un nivel de interfaz, es decir, solo definen las variables presentes en los mensajes de entrada y salida, y no definen los tipos de dato que las componen. Por ello, es necesario explorar los esquemas XML que han sido declarados o importados en la sección <wsdl:types> y de esta manera extraer la composición de tipos de dato que cada variable de una operación requiere para ser invocada.

Nuestro análisis de tipos se limita a extraer las definiciones de tipos primitivos, simples, complejos y de tipo elemento que se encuentren en los esquemas locales o importados que hagan referencia a otros documentos XML esquema externos y no a definiciones ontológicas. La definición de tipos

provista por XML esquema permite especificar la cantidad de apariciones de una misma variable así como su orden de aparición mediante los indicadores “Sequence”, “Choice”, “All”, “maxOccurs” y “MinOcurrs”. Sin embargo, nuestra representación registra cada tipo de dato como una entidad única, sin tomar en cuenta sus indicadores, para facilitar la representación de árboles y mantener un formato homogéneo.

3.1.2. Construcción de Árboles

La estructura de un documento XML está soportada por la organización jerárquica que poseen sus etiquetas. La jerarquización de etiquetas en documentos XML permite organizar información por categorías, las cuales derivan entre sí formando la estructura de un árbol [29].

Debido a que la naturaleza de un documento WSDL es la de pertenecer a una clase de documento XML, los archivos WSDL heredan la característica de representación de datos en una estructura de árbol (e.g. estructura listado 2.1). Por lo tanto, los nodos de un árbol de un archivo WSDL representan los elementos que componen a un SW.

La estructura de una operación se define por los tipos de dato que contiene y la organización de los mismos. Los tipos de dato complejo, simple o de tipo elemento de un documento XML Schema, encapsulan otras variables y proporcionan niveles de definición en la estructura general de una operación por tratarse de variables compuestas. Por lo tanto, la estructura de una operación posee una jerarquización no solo en la definición de los mensajes necesarios para su ejecución sino también en la definición de sus tipos de dato, es por esto que planteamos representar las operaciones de un servicio Web en árboles.

Proponemos representar las operaciones de SW en estructuras de datos que permitan realizar comparaciones semánticas y/o estructurales.

Aprovechando la estructura jerárquica que los archivos WSDL mantienen sobre la definición de cada elemento de un servicio Web, representamos operaciones en estructuras de árbol n-arias de forma que cada etiqueta de mensaje, operación o tipo perteneciente a la interfaz de una operación, represente a un nodo del árbol de la estructura de la operación Figura 3.1.

3.2. Representación RDF

La Web de datos o Web enlazada, describe un método para enlazar información estructurada por toda la Web. Compartiendo información de una manera que puede ser leída automáticamente por ordenadores. El formato estándar para la representación de datos en la Web enlazada es RDF.

Actualmente RDF es considerado un método general para la descripción conceptual o modelado de conocimiento, estructurando las relaciones entre los metadatos de un conocimiento. Sin embargo,

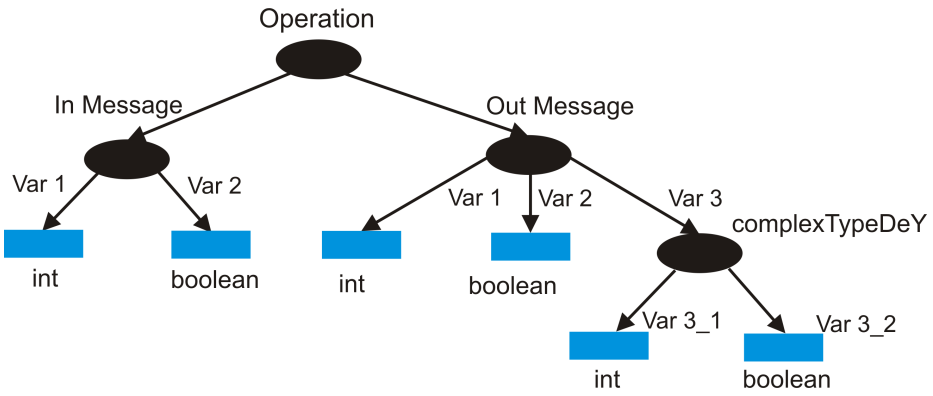


Figura 3.1: Árbol de una operación

RDF solo es un modelo de representación y su escritura en archivos legibles es llamada serialización. Existie una variedad de notaciones de sintaxis y formatos para la serialización de datos RDF.

Seleccionamos las serializaciones RDF como el formato de almacenamiento para las operaciones de servicios Web y para facilitar su adopción a la Web de datos, habilitando la construcción de una red de recomendación de servicios Web similares a través de la Web. El uso de serializaciones RDF permite además realizar inferencias lógicas para obtener análisis semánticos y proporcionar la estructura que tienen las relaciones de los metadatos para realizar comparaciones estructurales. Lo anterior permite habilitar el uso de métodos de comparación semántica y estructural de manera separada, para proporcionar flexibilidad en la elección o mejora de cada técnica sin afectarse entre ellas.

Proponemos representar las estructuras de árbol recuperadas de los archivos de definición WSDL en archivos RDF que permitan utilizar métodos semánticos en la comparativa de operaciones y que facilite la documentación automática basada en la reutilización de repositorios semánticos como DBpedia. Asociando el nombre de cada variable de operación con el archivo que contenga la definición semántica de cada una de ellas Figura 3.2.

Cada archivo RDF representa un grafo construido a partir de tripletas, las cuales pueden contener sujetos que tengan relaciones con objetos que solo representen información y con objetos que a su vez sean recursos RDF. De manera tal que la representación de conocimiento provista por un archivo RDF pueda reutilizare cuando se requieran establecer relaciones con conocimientos que han sido previamente desarrollados en archivos RDF. Es posible establecer relaciones entre diferentes recursos semánticos que enriquezcan un vocabulario especifico.

La Figura 3.2 muestra conexiones entre las variables de una operación y sus definiciones semánticas ubicadas en repositorios de conocimientos, construidos a partir de colecciones de archivos RDF que mapean información en relaciones de conocimiento.

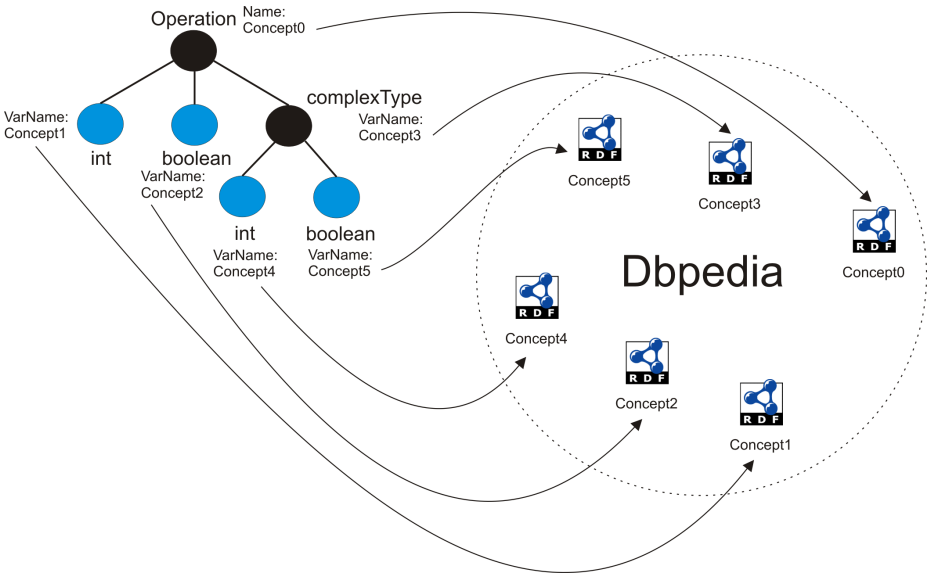


Figura 3.2: Documentación Automática

3.3. Comparación de Operaciones

Proponemos un método de correlación entre árboles para estimar la similitud estructural entre dos operaciones de servicios Web. Respetando la estructura de árbol que los documentos WSDL poseen en la definición de sus componentes. Construimos estructuras de árbol manipulables por el sistema comparador de operaciones y convertimos la estructura de cada operación descrita en el WSDL de un servicio Web en un árbol comparable.

Nuestra propuesta de comparación parte de la obtención de dos *árboles comparables*, mismos que son pesados en base a la cantidad de nodos que poseen y a los hijos que posee cada nodo de los árboles. Sin embargo, los *árboles comparables* aun no pueden ser comparados entre sí, debido a que poseen estructuras diferentes. Para normalizar dos *árboles comparables* y hacerlos compatibles para una comparación, los *árboles comparables* son transformados en árboles complementarios los cuales poseen nodos sin peso que acompletan la estructura del árbol comparable a una súper estructura llamada súper árbol, la cual contiene a los dos árboles que serán comparados. Finalmente la comparación se realiza entre los árboles complementarios de cada operación, de los cuales se extrae un vector característico formado por los pesos de cada árbol para después correlacionarlos y obtener un grado de similitud.

3.3.1. Comparación

Muchos algoritmos que estiman la similitud en estructuras de árbol implementan técnicas de comparación combinadas con métodos exhaustivos o combinatorios para determinar todas las simi-

litudes posibles entre los subárboles de dos árboles en específico. Algoritmos como "zhang & shasha distance" [30] y "Klein distance" [31] buscan la similitud de dos árboles basados en la lógica "Tree Edit Distance", buscando el número de modificaciones necesarias para igualar un árbol con otro y emitiendo un resultado en números de modificaciones. Los algoritmos basados en "Tree Edit Distance" calculan el número de modificaciones de similitud entre árboles mediante búsquedas de ruta más corta para cada modificación necesaria, lo que implica una cantidad considerable de procesamiento en la búsqueda de similitud para dos estructuras de árbol.

Proponemos un algoritmo de correlación estructural que no realiza búsquedas exhaustivas ni métodos combinatorios para correlacionar dos estructuras de árbol, con la finalidad de reducir el tiempo de procesamiento entre dos estructuras de árbol. El algoritmo propuesto, calcula un vector característico por cada árbol involucrado en una comparación, el cual contiene la información de la estructura de datos y permite su comparación.

La ventaja de utilizar métodos exhaustivos radica en la precisión que éstos pueden brindar en sus resultados a costa de mayor tiempo de cómputo. Nuestra propuesta realiza un compromiso entre precisión y rapidez la cual disminuye en menor medida la precisión para dar paso a una pronta obtención de resultados confiables.

El algoritmo se divide en cinco etapas, de las cuales cuatro realizan el tratamiento de las estructuras de dato a comparar y la quinta etapa de realizar la comparación de estructuras.

Etapas de tratamiento:

1. Obtención de pesos: Los árboles a comparar son examinados para obtener el número de hijos que cada nodo posee y el número de nodos asociados a cada sub árbol de las estructuras a comparar.

2. Ordenamiento: Reacomoda cada árbol en base a su peso, moviendo las subestructuras de mayor tamaño y separándolas de las subestructuras más pequeñas.

3. Obtención de árboles complementarios: Homogeniza ambos árboles de una comparación, para que estos posean la misma estructura y puedan ser comparados justamente.

4. Obtención de vectores característicos: Obtiene un vector numérico de la información estructural de cada árbol.

Etapa de comparación:

5. Correlación: Analiza los vectores característicos de la cuarta etapa para determinar la distribución de los mismos y realizar una corrección por cada nodo complementario encontrado para representar el valor real de la correlación. Finalmente, realiza una correlación de Pearson [32] entre los vectores corregidos, emitiendo un resultado de correlación en el rango [-1,1].

El funcionamiento del algoritmo requiere de la ejecución de las etapas anteriores en el orden de aparición.

3.3.1.1. Obtención de pesos

Es preciso establecer una convención numérica para especificar las relaciones existentes entre los nodos de un árbol y que a su vez permitan caracterizar a cada estructura de datos para su posterior comparación. Como parte de la caracterización de un árbol, proponemos dos medidas para representar la conexión entre nodos y la importancia que tiene cada subestructura del árbol.

La primera medida estructural que definimos es la de descendencia, la cual asocia el número de hijos que cada nodo de un árbol posee y representa la conexión entre nodos.

Con motivo de determinar la importancia de cada subestructura contenida en un árbol proponemos una segunda medida, la cual asocia un peso a cada nodo de un árbol. Para definir el peso de un árbol hacemos uso de la definición recursiva de una estructura de datos de árbol. Donde cada nodo de un árbol es a su vez un subárbol del mismo [33].

Finalmente, definimos el peso de un árbol o subárbol como la cantidad de nodos que éste posee incluyendo a su raíz.

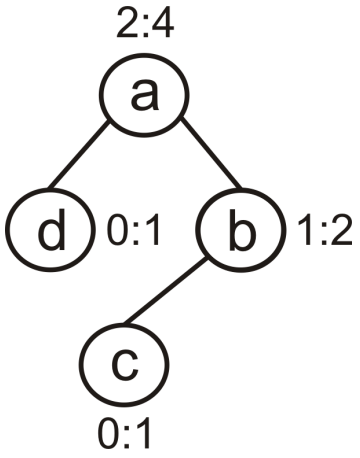


Figura 3.3: Árbol desordenado A

La Figura 3.3 Muestra una estructura de datos de árbol, la cual ha sido previamente pesada. La notación $H : P$ representa las medidas estructurales pertenecientes a cada nodo del árbol, donde H representa la cantidad de descendientes que un nodo posee y P el peso asociado al nodo o subárbol.

3.3.1.2. Ordenamiento

La etapa de ordenamiento es una parte fundamental en el algoritmo de correlación estructural que proponemos. El ordenamiento permite al algoritmo especificar un orden estructural común para

las estructuras de árbol. Permitiendo comparar árboles basándose en un acomodo que mantiene los nodos con mayor descendencia y peso a la izquierda de cada árbol.

Los criterios de ordenamiento permiten reacomodar el orden en el que se encuentran almacenados los nodos de un árbol, sin afectar las relaciones entre éstos. Desplazando las subestructuras de mayor relevancia a la izquierda del árbol a ordenar. Durante el ordenamiento de un árbol son considerados dos criterios de ordenamiento, por número de hijos y por peso.

El proceso de ordenamiento comienza por la raíz de cada subárbol contenido en la estructura. Ordenando a sus hijos en forma descendiente de izquierda a derecha, empleando como criterio principal la cantidad de hijos que cada nodo posee y utilizando como criterio secundario el peso asociado a cada nodo.

El algoritmo de ordenamiento para la figura 3.3 comienza por analizar la raíz representada por el nodo a y ordena a sus hijos d y b en base a la cantidad de hijos que estos poseen. El nodo b es intercambiado en su posición por la del nodo d ya que el nodo b posee un hijo y el nodo d no posee hijos. Recursivamente los nodos b y d son considerados como los nuevos árboles a ordenar sin embargo el nodo b solo posee un hijo el cual no es necesario ordenar y el nodo d no posee hijos a ordenar, por lo tanto finaliza el algoritmo dando como resultado el árbol de la figura 3.4.

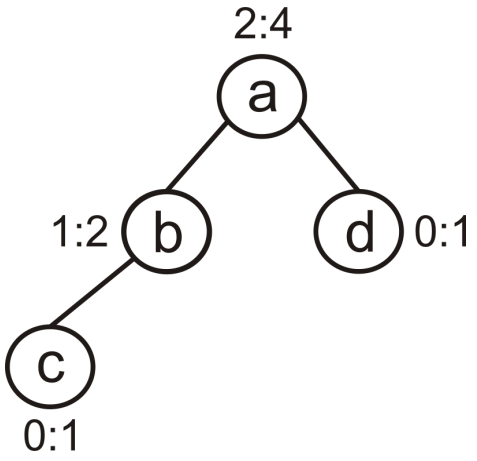


Figura 3.4: Árbol ordenado A

3.3.1.3. Obtención de árboles complementarios

Una comparación entre dos elementos cualquiera coteja las semejanzas existentes entre los atributos comunes de dichos elementos. Los atributos comunes entre dos o más objetos permiten establecer puntos de referencia para estimar similitudes entre objetos. Por lo anterior, es necesario imprimir un orden de comparación entre dos estructuras de árbol diferentes. Que permita comparar adecuadamente las conexiones entre los nodos de dos árboles distintos.

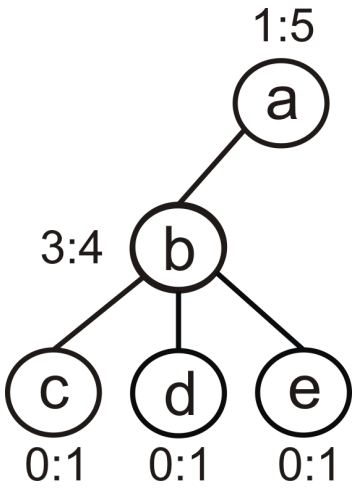


Figura 3.5: Árbol ordenado B

Para realizar la comparación entre dos árboles distintos, proponemos la creación de un árbol patrón el cual llamamos súper árbol y que contiene en su estructura a los dos árboles motivo de la comparación.

La relevancia de construir un súper árbol en nuestra propuesta, radica en la necesidad de completar cada árbol en la una comparación con el súper árbol resultante de la misma y de esta manera tener dos estructuras con diferentes pesos y que a su vez son comparables entre sí.

Para complementar los árboles de una comparación, estos son modificados agregando nodos complementarios a sus estructuras hasta obtener una configuración igual a la del súper árbol que rige a la comparación. Los nodos complementarios son nodos vacíos en sus medidas ya que no poseen hijos ni tampoco aportan peso a la estructura, por lo tanto poseen una medición $H : P$ de 0:-1.

La figura 3.6 representa el súper árbol de la comparación entre los árboles A y B, figuras 3.4 y 3.5. Las figuras 3.7 y 3.8 muestran los árboles complementarios para los árboles A y B.

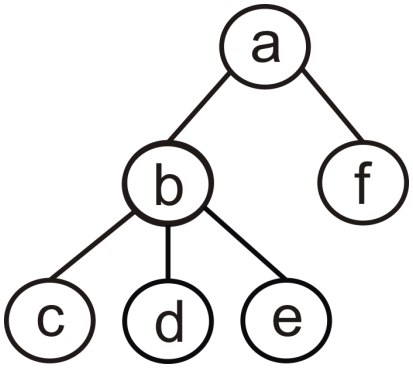


Figura 3.6: Súper árbol de la comparación entre los árboles A y B

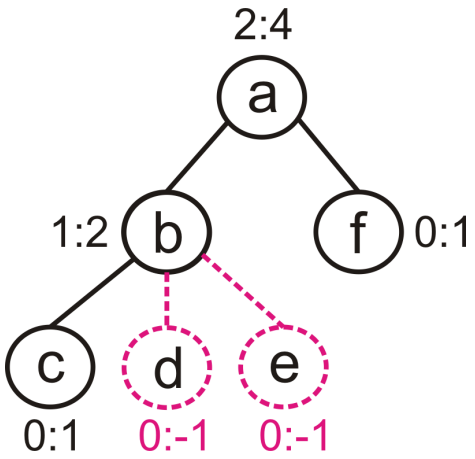


Figura 3.7: Árbol A complemento

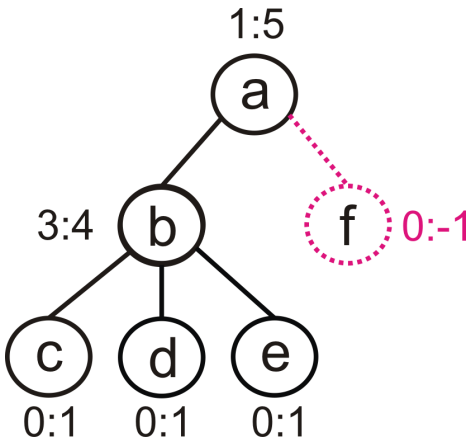


Figura 3.8: Árbol B complemento

3.3.1.4. Obtención de vectores característicos

Consideramos a la estructura de un árbol como la forma que establecen las relaciones de sus nodos. Existen dos tipos de relación que un nodo puede tener respecto a otros. Las relaciones de ascendencia y descendencia describen qué nodo es jerárquicamente superior a otro y qué nodos se encuentran en un mismo nivel.

Proponemos la obtención de un vector característico para extraer la información estructural de un árbol y poder compararla con la de otro. La construcción de un vector característico coloca en un arreglo la información sobre la estructura de cada nodo dentro de un árbol, es decir, la cantidad de descendientes que cada nodo posee. Adicionalmente, consideramos como información estructural el peso de cada nodo, el cual se compone de la cantidad de nodos que un árbol posee. Por lo tanto si consideramos la definición recursiva de un árbol, donde cada nodo es por sí mismo un árbol, cada sub árbol contenido en la estructura global posee un peso.

Finalmente, la construcción de un vector característico se realiza colocando dos valores consecutivos en el arreglo de información estructural por cada nodo existente en el árbol, es decir, la información sobre los descendientes del nodo y su peso. El orden de aparición de la información de cada nodo se apega a un recorrido en pre orden de la estructura del árbol, e.g. el vector característico para la [figuras 3.7 y 3.8] es la figura 3.9.

La obtención de vectores característicos para una comparación la restringimos a ser únicamente calculada en árboles complementarios, los cuales ya poseen el patrón del súper árbol que rige a la comparación. Permitiendo una comparación entre elementos estructurales equivalentes por la posición que estos guardan respecto a la estructura de su árbol.

3.3.1.5. Correlación de vectores característicos

Proponemos un método de correlación que hace uso de elementos estadísticos para el calculo de similitud entre estructuras de árbol, y de esta manera mantener tiempos de ejecución bajos en comparación con algoritmos de exhaustividad. Por tanto, es necesario comprender los conceptos de variables estadísticas, medias de centro y dispersión de datos enunciados en la sección 2.7 .

El algoritmo propuesto calcula la co-variación lineal entre dos vectores característicos pertenecientes a estructuras de árbol que requieren ser comparadas. Además de proporcionar el coeficiente de correlación de Pearson para indicar el grado de relación que existe entre ambos vectores característicos.

Los vectores característicos de los árboles complementarios de una comparación, se construyen a partir del recorrido en pre orden de cada árbol, almacenando en su vector la cantidad de hijos y el peso de cada nodo visitado durante el recorrido. De manera que la búsqueda de la co-variación entre vectores característicos se realice entre valores asociados a nodos que guardan una misma posición en el súper árbol que los contiene. Es decir, basados en las figuras 3.7 y 3.8 los vectores característicos generados A y B mostrados en la figura 3.9 ejemplifican que los pares pertenecen al mismo nodo.

A = {2, 4, 1, 2, 0, 1, 0, -1, 0, -1, 0, 1}
a b c d e f
B = {1, 5, 3, 4, 0, 1, 0, 1, 0, 1, 0, -1}

Figura 3.9: Vectores característicos de los vectores A y B

El contenido de dos vectores característicos como los ilustrados en la figura 3.9 puede ser sometido al cálculo de co-variación o covarianza, lo cual determinaría si los valores de cada par (A_i, B_i) son iguales o cercanos. Sin embargo, no reflejaría adecuadamente la diferencia estructural existente entre los nodos complementarios de un árbol y los no complementarios de otro árbol, debido a que todos los nodos complementarios poseen un peso de -1, lo que los agruparía en un solo punto de la gráfica

y evitaría la detección de incompatibilidad estructural al no presentar valores que modifiquen la dispersión de los valores bivariados y por consiguiente no reflejarían la co-variación lineal cuando existan diferencias estructurales.

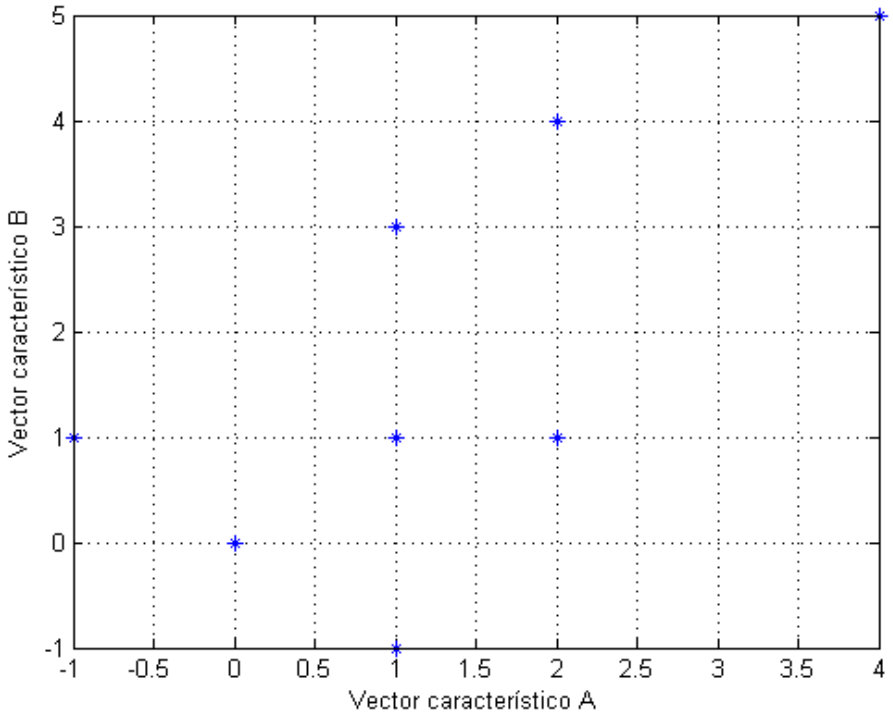


Figura 3.10: Grafica de dispersión para vectores característicos de A y B

La figura 3.10 representa la gráfica de dispersión para los vectores característicos de la figura 3.9 y muestra como los pares (-1,1) de los nodos *d*, *e* y *f* se grafican en un solo punto e impiden identificar la discrepancia estructural entre los nodos reales y los nodos complementarios al no mostrar una dispersión entre ellos que reduzca la similitud lineal del resto de los valores.

Proponemos un método de corrección para balancear la correlación de dos vectores característicos. Disminuyendo el factor de correlación por cada diferencia estructural encontrada, las cuales son detectables en un vector característico por cada valor de -1 en su contenido y que a su vez pertenecen a un nodo complementario dentro de las estructuras a comparar.

El proceso de balanceo para dos vectores característicos radica en construir un punto por cada diferencia estructural dentro de la gráfica de dispersión, que a su vez genere un patrón de relación negativo gradual al número de discrepancias estructurales. Contrarrestando el efecto de correlación positiva que la agrupación de pares con valores -1 ocasiona.

La correlación de Pearson (2.6) devuelve valores en el rango [-1, 1] y depende de la covarianza

(2.7) S_{xy} y de las desviaciones estándar (2.5) S_x y S_y de los vectores a correlacionar X y Y.

Cuando la correlación de dos vectores r_{xy} entrega valores cercanos al cero, indica que existe poca relación entre los vectores y de forma contraria cuando el resultado tiende a tomar valores cercanos a 1 y -1 se dice que existe una fuerte relación entre las variables.

Como es descrito en la sección 2.7 la correlación positiva o negativa de dos variables estadísticas depende de la distribución que sus valores posean respecto a la media de cada variable. Propiciando que distribuciones, cuyos datos bivariados, posean una recta ajustable de pendiente negativa genere correlaciones negativas. Es por esto que proponemos la construcción de una segunda recta ajustable con pendiente negativa para balancear la correlación entre las similitudes y las discrepancias estructurales existentes en los vectores característicos a comparar. Modificando cada par (A_i, B_i) generado por un nodo complementario y convirtiéndolo en un punto de la recta de pendiente negativa para compensar las discrepancias estructurales y así disminuir la correlación que dos vectores pudiésen tener. De manera que las correlaciones con resultados en el rango [-1,1] se interpreten de manera distinta. Interpretando valores de correlaciones de similitud nula a cualquier valor menor o igual a cero.

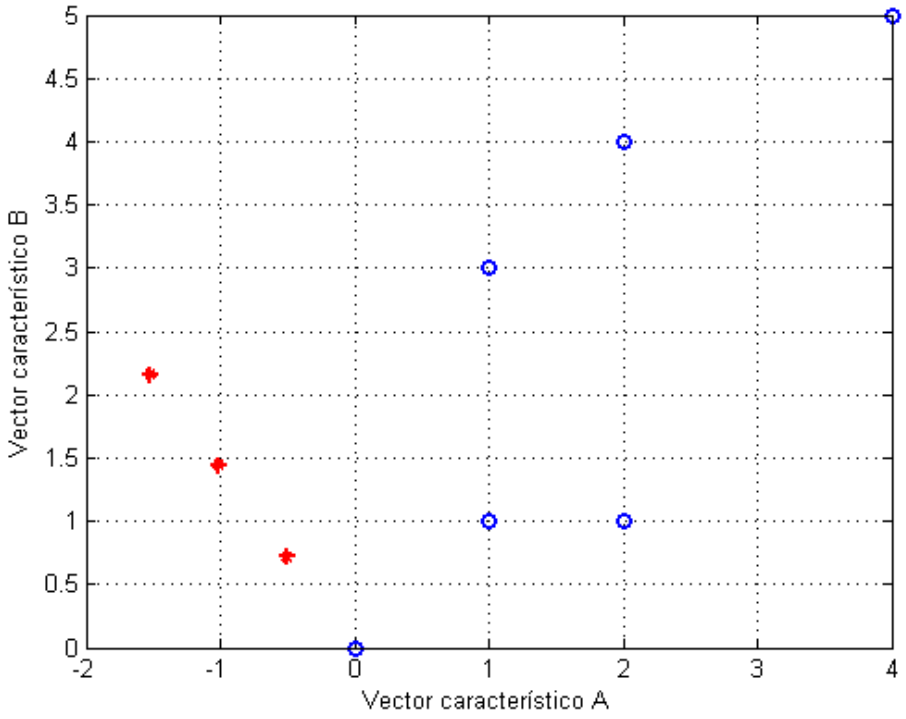


Figura 3.11: Grafica de dispersión para los vectores característicos A y B con recta de compensación

La figura 3.11 muestra la construcción de una segunda recta ajustable con pendiente negativa

para los vectores característicos de [figura 3.9] y así contrarrestar los efectos del agrupamiento de valores pertenecientes a nodos complementarios [figura 3.10].

Construir una pendiente negativa ajustable que contrarreste y estime la discrepancia estructural entre dos vectores característicos radica en parametrizar la distribución de datos que cada vector posee. Es de gran importancia conocer la información de la distribución para determinar el espaciado adecuado en los ejes vertical y horizontal de cada uno de los puntos de la compensación. Un espaciado incorrecto podría resultar en una dispersión inadecuada de la recta ajustable con pendiente negativa provocando correlaciones altas o bajas sin reflejar la similitud de los vectores característicos.

El espaciado adecuado entre los puntos de la recta de compensación es proporcional al espacio promedio existente entre los valores ordenados de cada eje de la gráfica de dispersión. Por lo anterior, se puede obtener el valor del espacio promedio en el eje X provisto por el vector A y el valor del espacio promedio en el eje Y provisto por el vector B.

Determinar el espacio promedio entre los puntos ordenados de un vector requiere de una ordenación implícita en los datos, lo que implicaría que la implementación de algún método de ordenamiento, incrementando el tiempo de cómputo en la obtención de un resultado. Para evitar un ordenamiento innecesario en la obtención del espacio promedio por cada vector, utilizamos la desviación estándar de los mismos. La desviación estándar nos proporciona la distancia promedio entre la media de un vector y cualquier elemento del vector [sección 2.7]. Por lo tanto, utilizamos dos veces la desviación estándar para inferir el rango [sección 2.7] de los datos, el cual es dividido entre el número de nodos no complementarios NNC que representan la dispersión de la gráfica, para de esta manera obtener el espacio promedio entre los puntos ordenados de un vector (3.1).

$$\begin{aligned} espaciado_x &= \frac{2S_x}{NNC} \\ espaciado_y &= \frac{2S_y}{NNC} \end{aligned} \quad ; NCC = \text{Número de nodos no complementarios} \tag{3.1}$$

Determinar el espacio adecuado entre cada punto de la recta de compensación, permite modificar los vectores característicos de A y B, reemplazando cada par (A_i, B_i) que contenga un valor de -1 por un par que represente un punto de la recta de compensación.

Definimos NC como el número total de nodos complementarios existentes en los vectores A y B. por lo tanto, existirán NC pares que serán reemplazados para construir la recta de compensación.

La construcción de cada par o punto de la recta de compensación está dada por,

$$(A_i, B_i) \longrightarrow (-k\frac{2S_x}{NNC}, k\frac{2S_y}{NNC}); k \in [1, 2, 3, ..., NC] \tag{3.2}$$

Modificando los vectores característicos A y B para reemplazar los nodos complementarios por puntos que en la gráfica de dispersión dibujen una recta con pendiente negativa apreciable en el

cuadrante de las ordenadas positivas y las abscisas negativas. Contrarrestando la dispersión del resto de los nodos durante la correlación de los vectores.

Finalmente, tras haber modificado los vectores característicos para construir una recta de compensación, se realiza la correlación entre dichos vectores. Calculando la correlación de dos vectores A y B haciendo uso de (2.6).

Resumen

Este capítulo presentó las etapas de la solución propuesta para realizar la comparación estructural entre dos operaciones de servicios Web diferentes. Comenzando por la extracción de las interfaces de cada operación descrita en los archivos de especificación WSDL y describiendo los elementos de relevancia para la obtención de la estructura de cada operación definida. Se puntualizaron los tipos de dato soportados por el lenguaje XML Esquema y su importancia en el análisis de los documentos WSDL para realizar una correcta extracción estructural por cada operación. Para facilitar la exploración y extracción de operaciones se utilizó el API Membrane SOA Model en su versión 1.4.1, cuyos elementos para la extracción fueron descritos.

Fueron presentadas las razones para mantener las estructuras operacionales en árboles no binarios, a la vez de ser descritas las clases Java desarrolladas para almacenar los elementos estructurales de las operaciones extraídas de las especificaciones de los servicios Web. Se describieron los métodos para recorrer, mostrar, serializar en RDF y comparar los árboles.

Explica las razones para utilizar archivos RDF como medios de almacenamiento para las operaciones de servicios Web. Exponiendo las facilidades para realizar nuevas comparaciones, proporcionar una separación entre las comparaciones semánticas y estructurales y finalmente permitir una adopción más sencilla a la web de datos.

Como parte fundamental explica el algoritmo propuesto para realizar comparaciones entre estructuras de árboles. Exponiendo la forma de recuperar la estructura de una serialización RDF y almacenarla en objetos del sistema para ser comparada. Se detalla el algoritmo propuesto para realizar la comparación entre árboles, utilizando una técnica de correlación o covarianza para disminuir la complejidad algorítmica en la comparación de dos estructuras de árbol y proporcionando una alternativa a los algoritmos clásicos de distancia editable para la comparación de estructuras de árbol.

Capítulo 4

Implementación

Como parte de las aportaciones del presente trabajo, el algoritmo de comparación estructural para operaciones de SW y el método de representación de operaciones en formato RDF fueron implementados en dos herramientas diferentes. Por un lado se desarrolló un API para que el algoritmo y el método de representación fuesen utilizados por cualquier aplicación Java y por otro lado se implementó un sitio Web para proporcionar una interfaz gráfica que facilite la interacción de un usuario con los elementos desarrollados de forma manual.

Tanto el API como el sitio Web fueron diseñados para permitir a usuarios y/o a sistemas realizar clasificaciones de SW y tomar decisiones en la construcción de servicios compuestos, así como habilitar la representación de operaciones en un formato semántico que permita la reutilización de operaciones por comparadores semánticos y posibilite la construcción de redes de recomendación de servicios mediante la conexión semántica de archivos RDF.

4.1. API

Las principales funcionalidades del API son: el parseo y la extracción de operaciones de archivos de especificación de SW (WSDL), la representación de operaciones en estructuras de árbol N-arias, la comparacion de estructuras N-arias y la representación de estructuras en archivos semánticos RDF.

4.1.1. Perseo y Extracción de operaciones

Para realizar la comparacion estructural de operaciones de SW se requiere analizar los archivos de definición WSDL que describen todos los componentes de un servicio Web. Mediante el análisis de archivos WSDL es posible identificar las definiciones de las operaciones que un servicio Web posee,

facilitando la extracción de la información referente a las mismas.

Una operación de un servicio Web al igual que una función o método codificado en algún lenguaje de programación, posee una interfaz de invocación y un valor de retorno. De manera análoga a un método programado en una aplicación de escritorio, las operaciones de servicios Web poseen mensajes de invocación y mensajes que definen el o los valores de retorno de una función. Por lo anterior, es necesario extraer la definición de tipos que compone a cada interfaz o mensaje de operación de servicio Web y de esta manera conocer la estructura de tipos necesaria en la invocación y recepción de resultados de una operación de servicio Web.

El objetivo de la extracción de operaciones en el presente trabajo pretende recuperar la definición jerárquica de tipos que describen las interfaces de las operaciones de los SW, mediante la lectura de los esquemas explícitos o importados de cada definición WSDL.

El desarrollo de este trabajo no se enfoca en recuperar exclusivamente la estructura que mantiene la definición de tipos de una operación, sino en recuperar tanto la estructura de las operaciones como toda la información que las define. Estableciendo como información relevante a los datos descriptivos de los tipos de datos, tales como sus nombres y los nombres de variable que los representan, los tipos de definición que tienen en el esquema de tipos, ya sean definiciones complejas (*complexType*), simples (*simpleType*), de elemento (*Element*) o primitivas tales como (`float`, `int`, `string`, `double`, etc.).

Para la extracción de operaciones necesitamos del mapa DOM que nos proporciona un índice para explorar cada elemento definido en los archivos WSDL. Es necesario recorrer el mapa de manera recursiva, debido a que se encuentra almacenado en un solo objeto Document. Durante el recorrido del mapa, exploramos las etiquetas de la tabla 3.1 y extraemos su información asociada.

La extracción y representación de estructuras de árbol se realiza de manera paralela. Cada vez que se encuentra una nueva operación en el mapa DOM, se crea un nuevo objeto de árbol n-ario y de igual manera, cada vez que se encuentra un elemento asociado a una operación, ya sea un mensaje de entrada y/o salida o un tipo de dato, se ingresa como nodo en la estructura de árbol asociada a su operación (figura 4.1).

Es importante recuperar toda la información asociada a cada elemento que compone a una operación para construir representaciones N-arias enriquecidas, que contengan toda la información indispensable para los algoritmos de comparación semánticos y estructurales, y a su vez permitan la extensibilidad del modelo de representación.

El mapa DOM proporciona un índice para explorar un documento WSDL, sin embargo no proporciona la información referente a los tipos de datos, por ello utilizamos el objeto Schema del API SOA Membrane en su versión 1.4.1, el cual contiene todas las definiciones de tipos locales o importadas encontradas en el documento WSDL. Para construir los nodos de una operación con la información de sus tipos de dato, construimos un almacén de tipos mediante tres tablas hash, que

almacenan todos los sub objetos de tipo Element, Simple y Complex contenidos en el objeto Schema, y asociar la información de tipo a cada nodo que se inserte en la representación N-aria.

Con motivo de diversificar los posibles alcances del proyecto se recuperan todos los datos antes descritos, para posteriormente almacenarlos en el formato de representación semántica RDF, habilitando su reutilización.

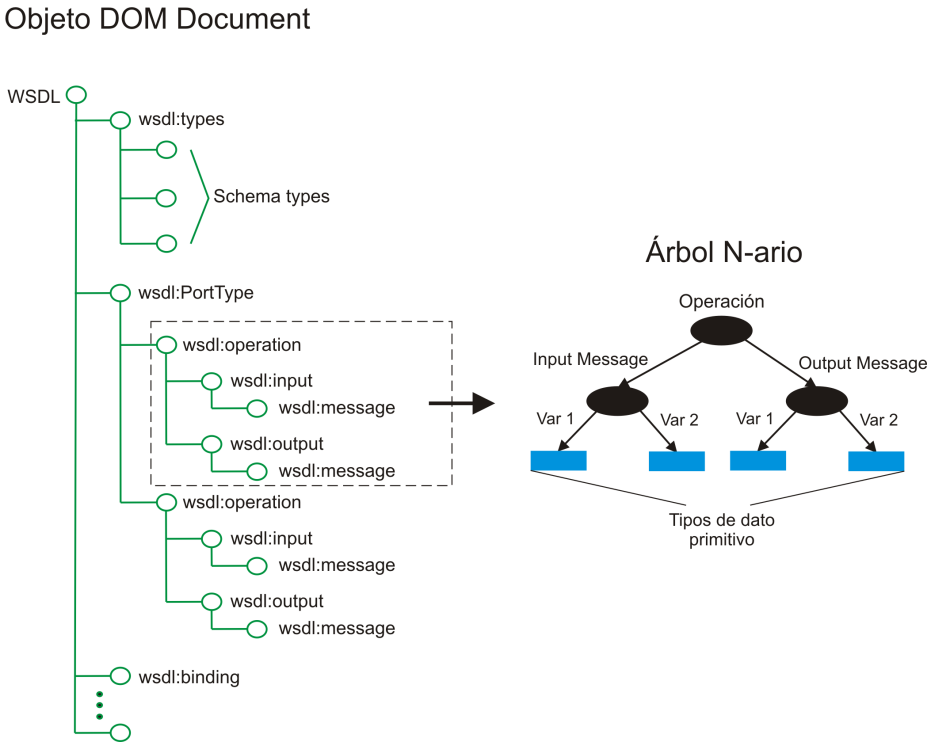


Figura 4.1: Estructura DOM de un archivo WSDL y mapeo de operación a estructura N-aria.

4.1.2. Construcción de árboles N-arios

La definición jerárquica natural de los archivos WSDL, mantiene organizadas las definiciones de los tipos de datos utilizados por las interfaces de las operaciones de servicios Web. Por lo cual, dichas definiciones ya se encuentran descritas en un formato de árbol N-ario. Sin embargo, la información devuelta por el parser SOA Membrane requiere ser adaptada a un formato manejable por las clases que realizan las comparaciones estructurales. Para ello, se desarrolló la clase `ArbolWSDL` del paquete `jnr.wsdltreestruct`, la cual encapsula las propiedades necesarias para el almacenamiento y comparación de las operaciones de un servicio Web.

La clase `ArbolWSDL` proporciona una estructura de árbol N-aria para almacenar la estructura y definición de tipos de una operación de servicio Web. Proporcionando los métodos necesarios para la inserción de datos y recorrido de información.

La construcción de las estructuras de árbol N-arias se realizó utilizando estructuras de tipo lista en Java, sub-contenidas y enlazadas entre sí. Esto, con la finalidad de brindar mayor control sobre la parametrización de los árboles y permitir la existencia de nodos con descendencia de N hijos.

Análogamente a un árbol real, los arboles poseen un tronco sobre el cual todas las ramas crecen, de igual manera nuestra estructura N-aria posee una lista principal la cual hacemos llamar el "tronco" de la estructura y nos indica todos los niveles de profundidad del árbol.

La lista principal o "tronco" se compone de sub-listas de nodos las cuales llamamos "ramas". Finalmente las hojas o nodos almacenados por las ramas pueden ser comparados con los nodos de cualquier árbol binario ya que estos tienen referencia directa hacia sus hijos. Sin embargo, en nuestra estructura N-aria un nodo no tiene referencia directa a otro nodo, sino con la rama que contiene a todos sus hijos (figura 4.2).

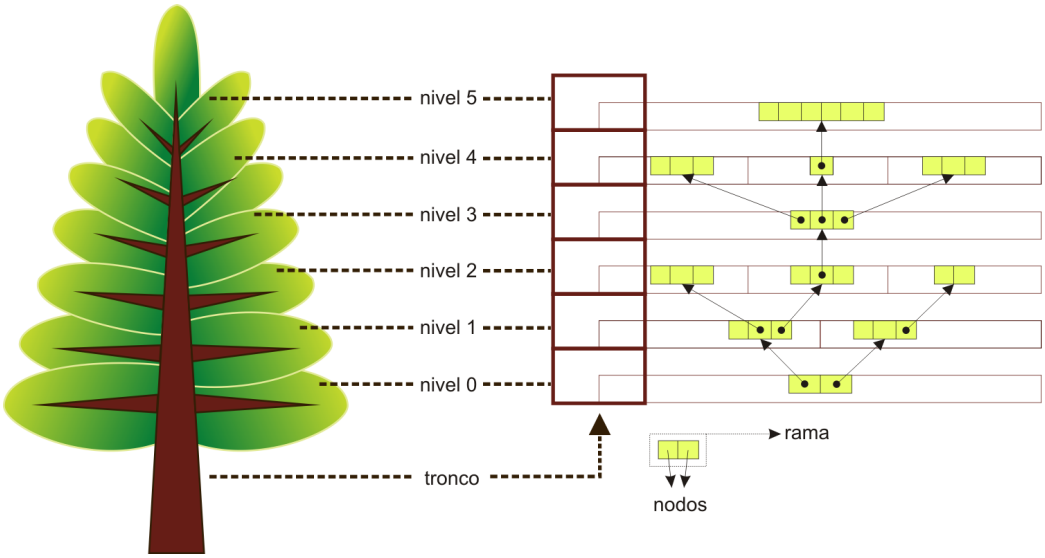


Figura 4.2: Construcción de árbol N-ario a partir de listas.

La razón de utilizar listas sub-contenidas y enlazadas entre sí (figura 4.2) para formar la estructura de árbol N-aria, radica en que las listas por sí solas son elementos dinámicos que crecen o decrecen fácilmente con la cantidad de elementos que contienen, manejando dinámicamente la memoria del programa. Adicionalmente, los métodos provistos por las listas permiten conocer la cantidad de elementos que contienen, facilitando la parametrización y pesaje de cada árbol dentro del algoritmo de comparación. Las listas además, permiten extraer sus elementos por medio del índice que poseen, lo cual ayuda a enlazar a los nodos de un árbol con la rama del siguiente nivel de profundidad en el "tronco" que contiene a sus hijos.

La figura 4.2 muestra la organización de las listas para formar el atributo tronco de la clase `ArbolWSDL` (figura 4.5). El tronco es una lista cuyos elementos son listas de ramas y donde el

índice de cada elemento representa la profundidad de los nodos del árbol. Las ramas se componen de nodos, los cuales a su vez pueden tener descendencia en una rama del siguiente nivel de profundidad, almacenando el índice que ocupa la rama descendiente cuando sea el caso.

Los nodos de un árbol almacenan la información de las variables de una operación de servicio Web. Cada nodo almacena la información de una variable en la forma en la que su tipo de dato es definido dentro del esquema XML del servicio Web, ya sea Complejo, Simple, Elemento o Primitivo. Elaboramos la interfaz `InterfazElementoWSDL` (figura 4.3) y la implementamos en la clase Nodo, de manera que dicha clase pudiese manejar correctamente cualquier definición de tipos descrita en las operaciones de un servicio Web y que ha sido almacenada en el atributo nodo de la clase. Las clases `ElementoXMLSchema` y `ElementoWSDL` proporcionan los atributos necesarios para almacenar los datos referentes a cualquier elemento WSDL o esquema relacionado con una operación de servicio Web.

La clase `ArbolWSDL` contiene los métodos necesarios para recorrer y mostrar la información de un árbol. Los métodos `imprimirArbol()` y `mostrarSerializacionRDF_xml()` imprimen en salida estándar la estructura del árbol almacenada por una instancia de la clase y la su representación RDF respectivamente.

4.1.3. Algoritmo de comparación

En el capítulo 3 se habló sobre el funcionamiento teórico del algoritmo de comparación estructural que esta tesis propone. En esta sección mostraremos de manera simple el funcionamiento del algoritmo a través del pseudocódigo de los métodos involucrados en el algoritmo de comparación estructural.

El algoritmo involucra la ejecución de 5 etapas diferentes:

1. **Obtención de pesos:** se parametrizan los árboles de una comparación; esta etapa utiliza los métodos `pesarArbol()` y `pesado()`.

2. **Ordenamiento:** se ordenan los nodos dentro de las ramas de los árboles a comparar, colocando en orden descendiente los nodos en base a su peso y número de hijos; esta etapa utiliza los métodos `ordenarArbol()` y `ordenarRama()`.

3. **Obtención de árboles complementarios:** crea un patrón entre los arboles a comparar, habilitando una comparación elemento a elemento entre arboles; esta etapa utiliza los métodos `obtenerArbolesComplementarios()` y generarSuperArboles.

4. **Obtención de vectores característicos:** extrae un vector con la informacion estructural representativa para el algoritmo; esta etapa utiliza el método `obtenerVectorCaracteristico()`.

5. **Correlación:** obtiene el coeficiente de correlación de Pearson para los vectores característicos de los árboles a comparar; esta etapa utiliza el método `correlacion()`.

Pseudocódigos

Algoritmo de comparación estructural

```
1:  comparacionEstructural(arbolA, arbolB)
2:     //Paso 1: Obtención de pesos
3:     arbolA ← pesarArbol(arbolA)
4:     arbolB ← pesarArbol(arbolB)
5:     //Paso 2: Ordenamiento
6:     ordenarArbol(arbolA)
7:     ordenarArbol(arbolB)
8:     //Paso3: Obtención de árboles Complementarios
9:     arbolesComplementarios[] ← obtenerArbolesComplementarios(arbolA, arbolB)
10:    //Paso 4: Obtención de vectores Característicos
11:    vectorA ← vacío
12:    vectorB ← vacío
13:    obtenerVectorCaracteristico(arbolesComplementarios[0].getRamaRaiz(), vectorA)
14:    obtenerVectorCaracteristico(arbolesComplementarios[1].getRamaRaiz(), vectorB)
15:    //Paso 5: Correlación
16:    return correlacion(vectorA, vectorB)
17: end
```

Algoritmo 1: Comparación estructural.

Etapa de obtención de pesos

El método principal en esta etapa es `pesado(rama)`, el cual realiza un recorrido recursivo sobre todos los nodos de un árbol. Obtiene los pesos asociados a cada nodo, partiendo del supuesto de que el peso de cada nodo es 1.

```
1:  pesarArbol(arbol)
2:     return pesado(hijo.getRamaDeHijos())
3:  end
4:  pesado(rama)
5:     pesoDelNodo
6:     pesoHijos ← 0
7:     hijos[] ← rama.getHijos()
```

```
8:    for each hijo de hijos do
9:       pesoHijos+=pesado(hijo.getRamaDeHijos())
10:   end for
11:   pesoDelNodo ← 1+pesoHijos
12:   return pesoDelNodo
13: end
```

Algoritmo 2: Obtención de pesos.

Etapa de ordenación

Debido que la estructura de los árboles N-arios se basa en la relación de listas subconetnidas, realizar la ordenación de los nodos solo requiere de ordenar los nodos de cada rama contenida en el árbol sin la necesidad de ordenar sus ramas. Por lo anterior el método `ordenarArbol()` manda a ordenar cada rama contenida en el tronco del árbol.

Por otra parte el método `ordenarRama()` ordena en forma descendiente a los nodos de una rama específica, colocando de izquierda a derecha los nodos con mayor relevancia estructural. Este método utiliza el algoritmo *QuickSort* para ordenar los nodos mediante dos criterios por peso y por número de hijos.

```
1:  ordenarArbol(arbol)
2:    tronco ← arbol.getTronco()
3:    for each rama en tronco do
4:       rama ← ordenarRama(rama)
5:    end for
6:  end
7:  ordenaRama(rama)
8:    nodos[] ← rama.getNodos()
9:    noNodos ← nodos.tamano()
10:   antiguosValores[3][noNodos]
11:   for posicion=0 hasta posicion < noNodos do
12:      antiguosValores[0][posicion] ← nodo[posicion].getPeso()
13:      antiguosValores[1][posicion] ← nodo[posicion].getNoHijos()
14:      antiguosValores[2][posicion] ← posicion
15:   end for
16:   //Ordenamiento Quick Sort,
17:   //primer criterio de ordenamiento: Peso,
18:   //segundo criterio: NoHijos
```

```
19:    nuevosValores ← quickSort(antiguosValores)
20:    //Ordenar el la nueva rama en forma desendiente
21:    nuevaRama ← vacía
22:    for i=noNodos-1 hasta  i >= 0  do
23:       antiguaPosicion ← nuevosValores[2][i]
24:       nuevaRama.insertarNodo(nodos[antiguaPosicion]
25:    end for
26:    return nuevaRama
27: end
```

Algoritmo 3: Ordenación.

Etapa de obtención de árboles complementarios

El métodos `obtenerArbolesComplementarios()` crea un árbol complementario para cada árbol referido en los parámetros de entrada. Construye el "súper árbol" que rige a la comparación y agrega nodos complementarios a los árboles para rellenar los espacios estructurales que les permitan parecerse al súper árbol de la comparación.

```
1:  obtenerArbolesComplementarios(arbolA, arbolB)
2:     superArboles[2]
3:     raizA ← arbolA
4:     raizB ← arbolB
5:     superArbolA ← vacío
6:     superArbolB ← vacío
7:     generarSuperArboles(SuperArbolA, SuperArbolB, raizA, raizB)
8:     superArboles[0] ← superArbolA
9:     superArboles[1] ← superArbolB
10:    return superArboles
11: end
12: generarSuperArboles(SuperArbolA, SuperArbolB, ramaA, ramaB)
13:    nodosA[] ← ramaA
14:    noNodosA ← nodosA.tamano()
15:    nodosB[] ← ramaB
16:    noNodosB ← nodosB.tamano()
17:    maximo ← max(noNodosA, noNodosB)
18:    for i=0 hasta  i  <=  maximo  do
19:       nodoA ← nodosA[i]
```

```
20:       rutaInsercionA ← nodoA.getUbicacion()
21:       nodoB ← nodosA[i]
22:       rutaInsercionB ← nodoB.getUbicacion()
23:       if nodoA tiene hijos y nodoB tambien then
24:          superArbolA.insertarNodo(Normal, rutaInsercionA+nodoA.getNombre())
25:          superArbolB.insertarNodo(Normal, rutaInsercionB+nodoB.getNombre())
26:          ramaA ← nodoA.getRamaDeHijos()
27:          ramaB ← nodoB.getRamaDeHijos()
28:          generarSuperArboles(SuperArbolA, SuperArbolB, ramaA, ramaB)
29:       end if
30:       if nodoA tiene hijos y nodoB no then
31:          superArbolA.insertarNodo(Normal, rutaInsercionA+nodoA.getNombre())
32:          superArbolB.insertarNodo(Complementario, rutaInsercionA)
33:          ramaA ← nodoA.getRamaDeHijos()
34:          generarSuperArboles(SuperArbolA, SuperArbolB, ramaA, null)
35:       end if
36:       if nodoB tiene hijos y nodoA no then
37:          superArbolA.insertarNodo(Complementario, rutaInsercionB)
38:          superArbolB.insertarNodo(Normal, rutaInsercionB+nodoB.getNombre())
39:          ramaB ← nodoB.getRamaDeHijos()
40:          generarSuperArboles(SuperArbolA, SuperArbolB, ramaB, null)
41:       end if
42:    end for
43: end
```

Algoritmo 4: Obtención de árboles complementarios.

Etapa de obtención de vectores característicos

El método `obtenerVectoresCaracteristicos()` construye un vector con la información de peso y número de hijos de cada nodo siguiendo un recorrido en pre orden sobre un árbol previamente parametrizado y ordenado.

```
1: obtenerVectorCaracteristico(rama, vector)
2:    nodos[] ← rama.getNodos()
3:    for each nodo de nodos[] do
4:       vector.agregar(nodo.getPeso())
5:       vector.agregar(nodo.getNoHijos())
```

```
6:      if nodo tiene hijos then
7:          obtenerVectorCaracteristico(nodo.getRamaDeHijos(), vector)
8:      end if
9:  end for
10: end
```

Algoritmo 5: Obtención de vectores característicos.

Etapa de correlación

El algoritmo de correlación adecua los vectores característicos para construir una correlación negativa de magnitud proporcional al número de nodos complementarios existentes en los vectores (Sección 3.3.1.5). Finalmente, calcula la correlación de Pearson en base a los vectores característicos modificados.

```
1:  CORRELACION(vectorA, vectorB)
2:      tamano ← vectorA.tamano()
3:      nodosNoComplementarios ← 0
4:      //Contar nodos no complementarios
5:      for i=0 hasta  i < tamano  do
6:          if vectorA[i] > 0 y vectorB[i] > 0 then
7:              nodosNoComplementarios += 1
8:          end if
9:      end for
10:     //Calculando tamaño de paso para compensación
11:     pasox ← 2 * desStd(vectorA)/nodosNoComplementarios
12:     pasoy ← 2 * desStd(vectorB)/nodosNoComplementarios
13:     //Compensando vectores
14:     nodosComplementarios ← 0
15:     for i=0 hasta  i < tamano  do
16:         if vectorA[i] < 0 o vectorB[i] < 0 then
17:             nodosComplementarios += 1
18:             vectorA[i] ← (-1)*(nodosComplementarios*pasox)
19:             vectorB[i] ← nodosComplementarios*pasox
20:         end if
21:     end for
22:     //Factor de correlacion
23:     return covarianza(vectorA, vectorB)/ (desStd(vectorA)*desStd(vectorB))
```

24: **end**

Algoritmo 6: Correlación.

4.1.4. Representación de operaciones en formato RDF

Para realizar la comparación de SW, se requiere forzosamente analizar los archivos de especificación de los servicios a comparar, con la finalidad de extraer los elementos relevantes para un determinado método de comparación. Por lo anterior, parsear los archivos de especificación WSDL es un trabajo obligado por los interesados en estimar similitudes entre servicios. Independientemente del tipo de comparativa ya sea semántica o estructural, es necesario parsear un documento para extraer la información distintiva de un servicio y así permitir su comparación. Es por esto que, como parte de nuestra aportación creamos representaciones semánticas de las operaciones de servicios Web para permitir la reutilización de operaciones por parte de otras aplicaciones. Proporcionando estructuras de árbol listas para ser utilizadas por cualquier algoritmo de comparación semántico o estructural, sin la necesidad de parsear nuevamente las especificaciones WSDL. Para ello, representamos las estructuras de árbol recuperadas en archivos RDF, mismos que habilitan el uso de métodos semánticos para la comparativa de operaciones y facilitan la integración de métodos capaces de organizar semánticamente a las operaciones de servicios Web.

Para convertir los arboles N–arios en serializaciones RDF, utilizamos el api apache Jena en su versión 2.11.1 la cual proporciona las clases necesarias para construir serializaciones RDF en diversos lenguajes. Utilizamos XML como el lenguaje predeterminado para las serializaciones RDF de los arboles N-arios, mismo que indicamos al API Jena para construir documentos XML válidos para representar cada operación en RDF.

Jena es utilizado en los métodos `getRDFModel()` y `almacenarSerializacionRDF_xml()` en la clase `ArbolWSDL` para construir las representaciones semánticas de cada árbol.

Para serializar cada árbol N-ario en archivos RDF-XML, es necesario almacenar la estructura de un árbol y la información de cada nodo en tripletas. Para representar la relación de los nodos de un árbol mediante tripletas, los sujetos de cada tripleta son asignados a cada nodo del árbol, los predicados representa la el tipo de relación entre un nodo y otro, ya sea relaciones de servicio, mensaje, operación y tipo. Finalmente, los objetos de cada tripleta, son a su vez los sujetos de otras tripletas, donde cada objeto de un sujeto representa a su hijo en la estructura del árbol (figura 3.1). Se creó la clase `RDFVocabulary` para almacenar etiquetas predeterminadas y a través de estas establecer los predicados de las relaciones RDF.

4.1.5. Descripción de clases

Paquete `datatypeextraction`, extracción y representación de datos (figuras 4.3, 4.4).

Paquete `Wsdltreestruct`, estructura de arboles (figura 4.5).

Paquete `operationmatcher`, comparación estructural (figura 4.6).

Paquete `utilities`, utilidades (figura 4.7).

Clase	Descripción
	Paquete: `Wsdltreestruct`
`ArbolWSDL`	Esta clase estructura la información de una operación de servicio Web, organizándola y almacenándola en un árbol N-ario. Proporciona los métodos necesarios para la creación, recorrido y comparación de árboles. Posee métodos de clase que incorporan el algoritmo de comparación estructural que propone esta tesis. Dichos métodos estáticos permiten la comparación directa entre instancias de esta clase.
`Nodo`	Representa la unidad fundamental de los arboles N-arios, ya que sus instancias almacenan la información recuperada de cada elemento de una operación de servicio Web. Almacena la información en una instancia de la clase `ElementoWSDL` la cual hereda de la clase `ElementoXMLSchema`, pudiendo representar definiciones de tipo de dato o propias de los mensajes de entrada y salida definidas en los archivos WSDL.Implementa la interfaz `InterfazElementoWSDL` para poder mostrar su información, independientemente del tipo de clase utilizada en la representación de información.Dentro de un árbol N-ario, un nodo puede tener descendencia de N hijos, por lo cual, esta clase posee un atributo que almacena el identificador de la rama que contiene a sus hijos.

`Rama`	Esta clase almacena a todos los hijos de un nodo. Se compone de los atributos necesarios para identificarse en una estructura de árbol N-ario. Almacena el índice de la rama que contiene al nodo padre de una instancia de la clase. Posee los métodos necesarios para la inserción y recuperación de nodos en la rama.
Paquete: `datatypeextraction`	
`AtributoXML`	Permite almacenar individualmente los atributos de una etiqueta XML ya sean de una definición de esquema o de cualquier etiqueta WSDL. Permitiendo almacenar la mayor información posible y poderla utilizar en las representaciones semánticas de operaciones.
`ElementoWSDL`	Esta clase permite representar únicamente a cualquier elemento WSDL que defina a una operación de un servicio.
`ElementoXML`	Esta clase almacena individualmente y de manera genérica a las etiquetas XML encontradas en los archivos de especificación de servicios, almacenando el nombre de cada etiqueta y sus atributos asociados mediante una lista de elementos AtributoXML.
`ElementoXMLSchema`	Esta clase permite representar únicamente a los elementos de definición de tipos de dato dentro de la sección *<wsdl:types>* de un documento WSDL.
`InterfazElementoWSDL`	Proporciona los métodos abstractos para la interacción con objetos de la clase `ElementoXMLSchema` y `ElementoWSDL`.
`WSDLExtractStructure`	Esta clase encapsula todos los métodos necesarios para analizar y extraer las operaciones de servicios Web, mediante el parseo de archivos WSDL. Estructura la información de operaciones y la almacena estructuras de árboles N-arios, tantas como operaciones poseea un servicio Web.
Paquete: `operationmatcher`	
`MatcherResult`	Proporciona instancias independientes para almacenar los resultados de comparación entre operaciones provenientes de la ejecución de los métodos de la clase `TreeMatcher`.

`RDFVocabulary`	Contiene un vocabulario predefinido para construir las relaciones semánticas de las operaciones de servicios Web y a través de ellas construir sus serializaciones RDF.
`TreeMatcher`	Proporciona métodos generalizados para la extracción y comparación automática de operaciones de Servicios Web, además de poseer un método para la conversión de operaciones a formato RDF. Los métodos permiten comparar operaciones, partiendo del tipo de almacenamiento que poseen las especificaciones WSDL, ya sean en una dirección local o en una URL de la Web. Adicionalmente, ofrecen diferentes modalidades de extracción y comparación las cuales son:Extracción y comparación entre las operaciones de dos archivos WSDL, entre todas las operaciones pertenecientes a los archivos WSDL de un repositorio y entre las operaciones de un solo archivo WSDL y las de un repositorio. Esta clase posee un método para extraer todas las operaciones de un repositorio y convertirlas en archivos RDF individuales.
Paquete: `utilities`	
`CompresionDeDatos`	Esta clase contiene métodos para comprimir archivos en un formato ZIP, mismo que es utilizado por el sitio Web para entregar las conversiones operación-RDF.
`Directorio`	Contiene métodos para explorar directorios locales y direcciones URL. Esta clase es utilizada en el análisis de directorios para la recuperación de los archivos WSDL.
`Estadistica`	Contiene métodos utilizados por el algoritmo de comparación propuesto.

Cuadro 4.1: Clases del proyecto.

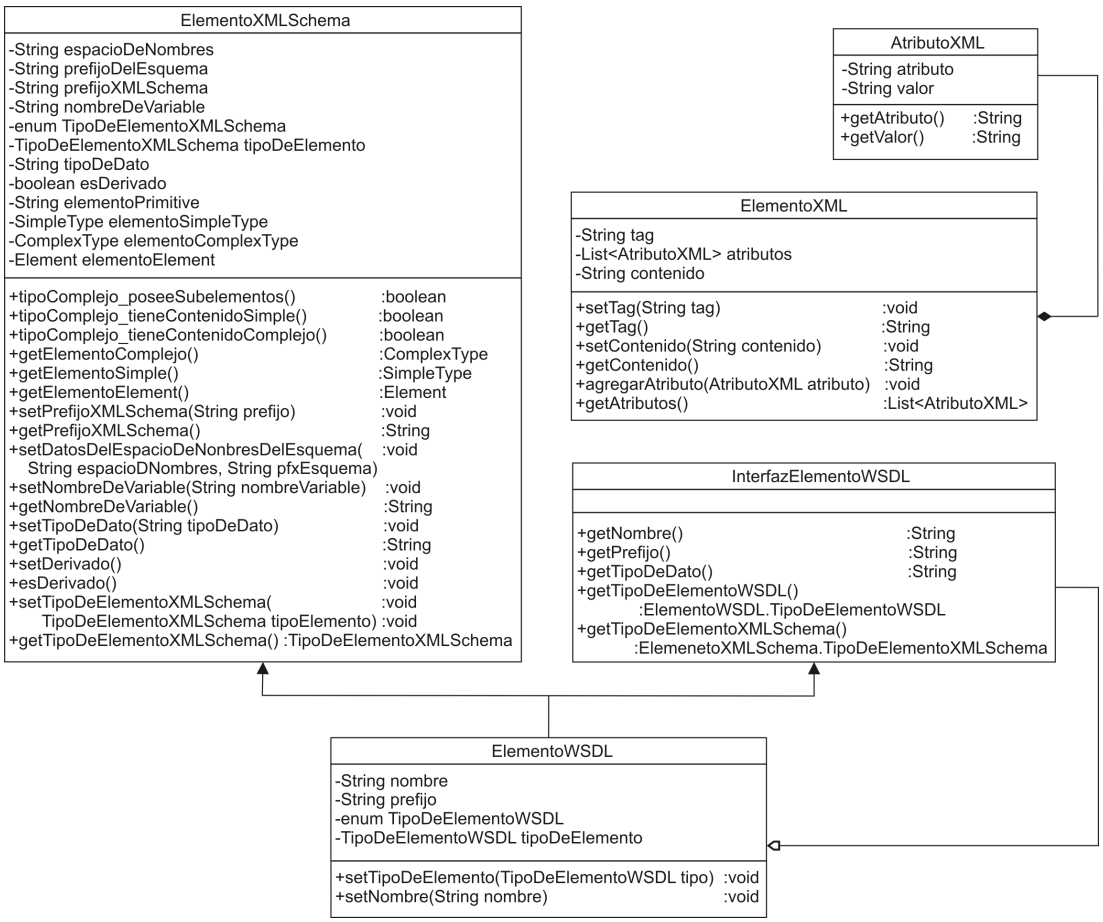


Figura 4.3: Diagrama de clases del paquete de extracción y representación de datos.

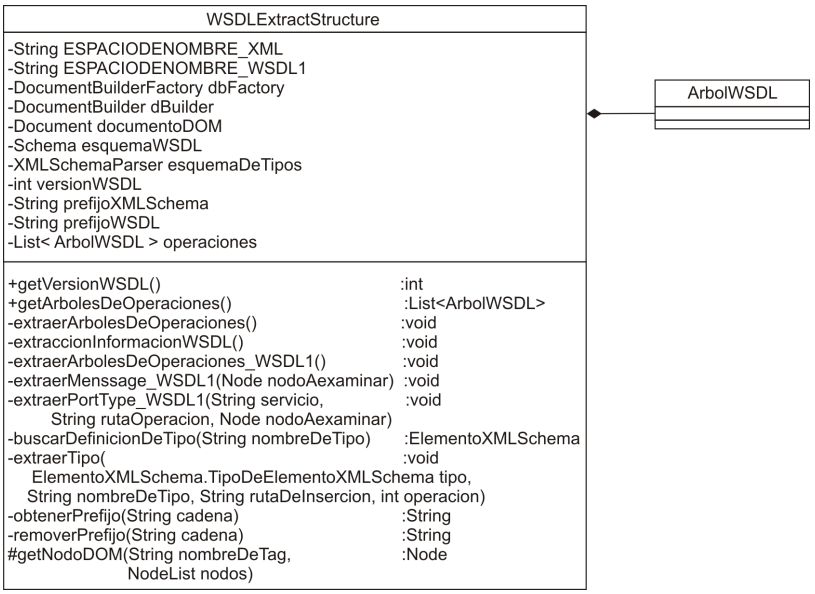


Figura 4.4: Diagrama de clases del paquete de extracción y representación de datos.

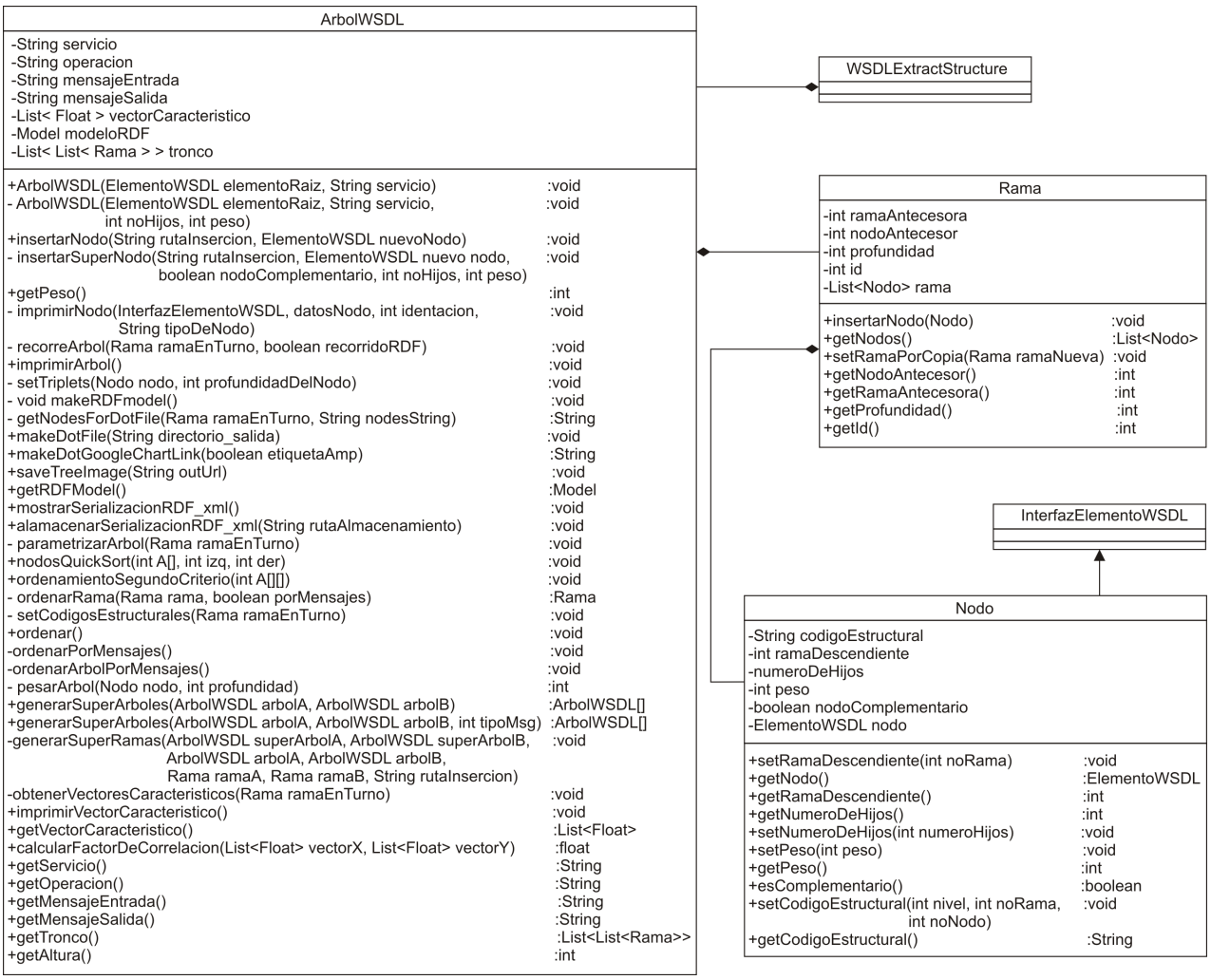


Figura 4.5: Diagrama de clases del paquete de estructura de árboles.

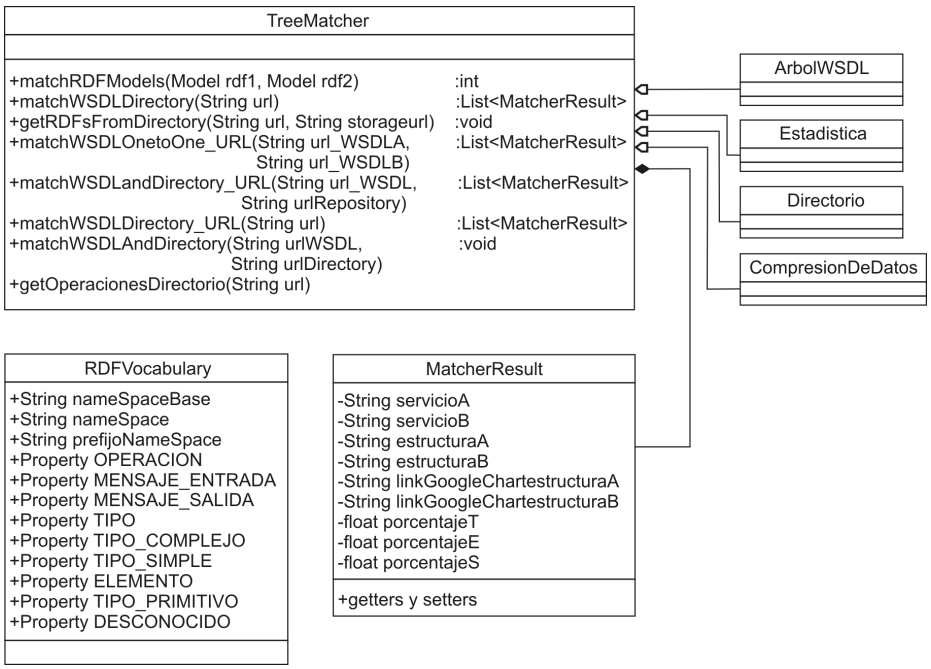


Figura 4.6: Diagrama de clases del paquete de comparación estructural.

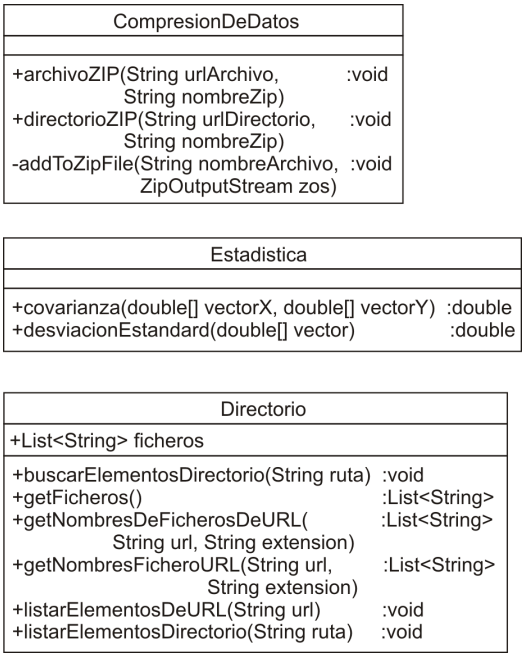


Figura 4.7: Diagrama de clases del paquete de utilidades.

4.1.6. Requerimientos del API

Se requiere la inclusión de las siguientes dependencias para el correcto funcionamiento del API.

- **Apache Jena 2.11.1**[1]**:** creación y validación de archivos RDF.
- **Jsoup 1.7.3**[2]**:** exploración de directorios y URLs.
- **SoaModel 1.4.1**[3]**:** Análisis y extracción de datos en archivos WSDL.

4.2. Sitio Web

El desarrollo del sitio Web se realizó mediante la construcción de una página Web dinámica codificada en el lenguaje de programación Java, lo cual permitió integrar el API del proyecto de tesis y utilizar sus funcionalidades. El sitio Web utiliza la tecnología JSF (por sus siglas en ingles Java Server Faces)[34] para construir la interfaz de usuario del sitio Web, para ello utiliza el Framework PrimeFaces en su versión 3.5 que imprime un modelo vista controlador a la construcción del sitio Web.

El sitio permite realizar comparaciones estructurales entre operaciones de servicios Web, partiendo de sus especificaciones WSDL. Las versiones WSDL soportadas por el sito son 1.x. Además, proporciona un conversor de operaciones WSDL a formato RDF.

El sitio web requiere de las siguientes dependencias para ejecutarse:

- **PrimeFaces 3.5**[4]**:** Framework de los controles de la interfaz de usuario.
- **iText-2.1.7**[5]**:** Permite exportar la información de las tablas en archivos PDF.
- **poi-3.7**[6]**:** Permite exportar la información de las tablas en archivos XLS.
- **commons-fileupload-1.2.1**[7]**:** Permite subir archivos al sitio Web.
- **commons-io-1.4**[8]**:** Permite subir archivos al sitio Web.

[1]Apache Jena, Página principal del framework, https://jena.apache.org
Octubre 2014
[2]Jsoup, Página principal, http://jsoup.org/
Octubre 2014
[3]SoaModel, Página principal del API, http://www.membrane-soa.org/soa-model/
Octubre 2014
[4]PrimeFaces, Página principal del framework, http://primefaces.org/
Octubre 2014
[5]iText, Página principal del proyecto iTextPDF, http://itextpdf.com/
Octubre 2014
[6]Poi, Página principal del proyecto Poi de Apache, http://poi.apache.org/
Octubre 2014
[7]commons-fileupload, Página principal del proyecto commons-fileupload de Apache, http://commons.apache.org/proper/commons-fileupload/
Octubre 2014
[8]commons-io, Página principal del proyecto commons-io de Apache, http://commons.apache.org/proper/commons-io/
Octubre 2014

Capítulo 5

Resultados

Este capítulo presenta las pruebas realizadas a la correlación de operaciones de servicios Web a través de la aplicación Web desarrollada. Las pruebas consistieron en la comparación estructural de las operaciones pertenecientes a 50 servicios Web almacenados en un repositorio de servicios multipropósito. Se obtuvo el nivel de correlación entre cada par de operaciones comparadas por el sistema, mismas que fueron contrastadas con las mismas comparaciones hechas por individuos y medidas con el uso de la métrica "precision & recall", la cual mide la claidad de los resultados del algoritmo propuesto.

5.1. Selección de metrica

El algoritmo de comparación estructural propuesto en esta tesis se puede parametrizar en dos aspectos diferentes, en el tiempo de ejecución para realizar comparaciones y en la efectividad del algoritmo para arrojar resultados coherentes y confiables. Sin embargo, sólo se decidó hacer pruebas para medir la efectividad y confiabilidad de los resultados, ya que no existen condiciones suficientes para realizar mediciones de rendimiento de ejecución. Para medir adecuadamente el rendimiento de ejecución, es necesario proporcionar estructuras de árbol de tamaños considerables, que permitan estresar al algoritmo de comparación y forzar su desempeño en la región de mayor complejidad algorítmica. Los árboles de definición para una operación de servicio Web promedio poseen una cantidad pequeña de nodos y de entre tres a seis niveles de profundidad. Tomando en cuenta que la definición de una operación posee una estructura fija para los primeros dos niveles (figura 5.1, las estructuras de árbol no poseen las ramificaciones suficientes en los siguientes niveles de profundidad como para construir árboles que representen una carga real para el algoritmo y permitan medir su rendimiento.

Para realizar las pruebas de confiabilidad sobre los resultados del algoritmo, se decidió utilizar

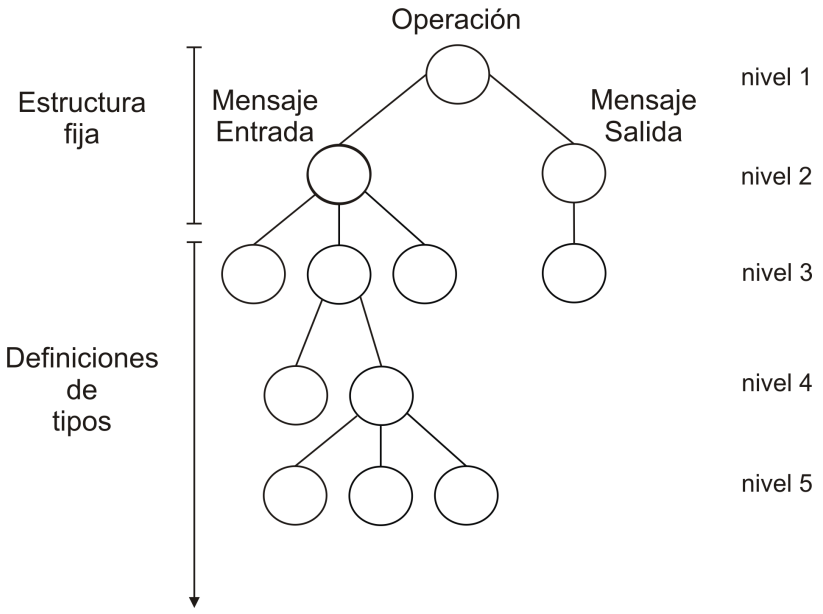


Figura 5.1: Estructura promedio de una operación de servicio Web.

la métrica "Precision and Recall", la cual permite medir el rendimiento de algoritmos de búsqueda, recuperación de información y reconocimiento de patrones.

5.2. Métrica Precision and Recall

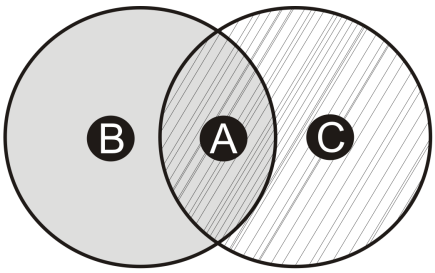
La métrica *Persicion and Recall* [35] requiere de un conjunto X de elementos sobre los cuales se efectuara una búsqueda, un patrón P de búsqueda sobre los elementos del conjunto X y un subconjunto Y del conjunto X que representa a los elementos relevantes o de mayor similitud con P que deberían de aparecer tras la búsqueda del patrón P en el conjunto X.

Después de efectuar la búsqueda del patrón P sobre el conjunto X, el algoritmo arrojara tres subconjuntos A, B y C (figura 5.2), donde el subconjunto A representa a todos los elementos relevantes recuperados por el algoritmo, el subconjunto B a los elementos relevantes no recuperados y el subconjunto C a los elementos irrelevantes que han sido devueltos por el algoritmo como similares al patrón P. De manera que: $X = A + B + C$ y $Y = A + B$.

Precision: Es la tasa de elementos relevantes recuperados en relación con la cantidad de elementos nos relevantes recuperados.

$$Precision = (A/(A + C)) * 100 \tag{5.1}$$

Recall: Es la tasa de elementos relevantes recuperados en relación con la cantidad total de elementos relevantes.



A: Conjunto de elementos relevantes recuperados
B: Conjunto de elementos relevantes no recuperados
C: Conjunto de elementos irrelevantes recuperados

Figura 5.2: Conjuntos importantes para la métrica “precision and recall”.

$$Recall = (A/(A + B)) * 100 \tag{5.2}$$

5.3. Conjunto de Pruebas

Realizar la comparación de dos objetos cualesquiera requiere de la detección de los elementos en común entre dichos objetos, y un criterio de comparación que permita estimar la similitud entre los objetos.

Debido a que existe una gran variedad de enfoques en la comparación de servicios Web y a que a cada enfoque requiere diferentes parámetros para realizar una comparativa, las evaluaciones en rendimiento y efectividad son hechas mediante conjuntos de pruebas construidos por selecciones manuales y son evaluados confrontando comparaciones humanas con los resultados de cada enfoque. Por lo anterior, las pruebas realizadas se efectuaron haciendo mediciones que contrastaran los resultados del algoritmo con los resultados obtenidos por comparaciones humanas.

Para utilizar la métrica “Precison and Recall” sobre el algoritmo de comparación, se estableció un escenario de pruebas con 50 operaciones de servicios Web como el conjunto X de las comparaciones y una operación extra como el patrón P comparar.

Se realizaron varias comparaciones entre el conjunto de operaciones X y el patrón P, una de ellas se realizó con el algoritmo de comparación y el resto de ellas fueron hechas por diferentes personas que en base a sus criterios establecieron las similitudes entre las estructuras de árbol del conjunto X y el patrón P. Las comparaciones provistas por las personas fueron tomadas como confiables y en base a ellas se contrastaron los resultados entregados por el algoritmo.

Debido a que el algoritmo propuesto sólo realiza comparaciones estructurales entre árboles de operaciones y no compara la semántica que éstos poseen en la definición de sus variables o tipos de datos, las comparaciones de prueba no requieren de servicios similares en su contexto de uso.

Por lo tanto, el conjunto de pruebas se compone de operaciones aleatorias que no necesariamente poseen una correlación en su semántica, ya que para el algoritmo de comparación estructural sólo son necesarios los árboles de las operaciones del conjunto de pruebas.

La tabla A.1 del apéndice A muestra las operaciones que componen el conjunto de pruebas:

5.4. Resultados

Se pidió la ayuda de cuatro personas para realizar las mismas comparaciones realizadas por el algoritmo. Se les proporcionaron las imágenes de los árboles de a cada operación del conjunto de pruebas. Las imágenes ejemplifican los arboles de cada operación y dentro de sus nodos contiene una etiqueta que indica el tipo de elemento que contiene cada nodo (figura 5.5).

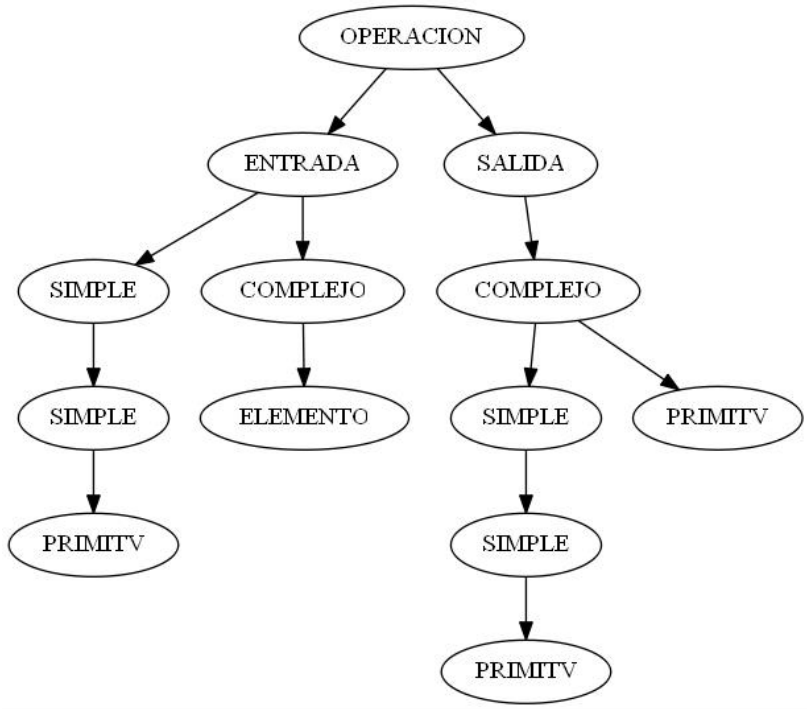


Figura 5.3: Tipo de estructura utilizada para las comparaciones humanas.

Para poder medir la efectividad del algoritmo ante la comparación de estructuras, se pido a los voluntarios realizar dos tipos de comparaciones, una comparación meramente estructural cuyos resultados contiene el cuadro 5.1 y una segunda comparación la cual toma en cuenta el contenido de los nodos. La segunda comparación (cuadros 5.2, 5.3), evalúa la similitud entre los mensajes de entrada y salida y entre el árbol completo de cada operación.

		Algoritmo	Sujeto 1		Sujeto 2		Sujeto 3		Sujeto 4	
Operación	nodos	Comparación	Se parece	%	Se parece	%	Se parece	%	Se Parece	%
1	13	1.00	1	1.00	1	1.00	1	1.00	1	1.00
2	9	0.95	1	0.65	1	0.65	1	0.60	1	0.65
3	6	0.60	0	0.40	0	0.35	0	0.45	0	0.40
4	6	0.11	0	0.30	0	0.15	0	0.20	0	0.30
5	6	0.60	0	0.40	0	0.35	0	0.42	0	0.35
6	32	-0.39	1	0.55	0	0.15	0	0.40	0	0.40
7	31	0.60	0	0.45	1	0.70	1	0.50	1	0.55
8	8	0.88	1	0.70	1	0.65	1	0.65	1	0.60
9	17	0.86	1	0.75	1	0.85	1	0.75	1	0.75
10	15	0.91	1	0.80	1	0.95	1	0.85	1	0.85
11	16	0.62	1	0.85	1	0.70	1	0.70	1	0.65
12	4	-0.04	0	0.30	0	0.35	0	0.40	0	0.35
13	24	0.59	1	0.80	1	0.55	1	0.50	1	0.65
14	31	-0.36	1	0.55	0	0.35	1	0.50	0	0.45
15	6	0.60	0	0.40	0	0.35	0	0.45	0	0.40
16	10	0.77	1	0.70	1	0.65	1	0.55	1	0.65
17	18	0.78	1	0.65	1	0.60	1	0.60	0	0.50
18	14	0.72	1	0.85	1	0.90	1	0.70	1	0.75
19	6	0.60	0	0.35	1	0.70	0	0.40	1	0.55
20	5	0.36	0	0.30	1	0.50	0	0.35	0	0.40
21	34	-0.44	1	0.55	0	0.45	0	0.45	0	0.40
22	6	0.60	0	0.35	1	0.65	1	0.60	0	0.45
23	11	0.87	1	0.95	1	0.85	1	0.70	1	0.85
24	15	0.87	1	0.75	1	0.65	1	0.75	1	0.75
25	8	0.81	0	0.45	1	0.90	1	0.80	1	0.90
26	17	0.74	1	0.60	0	0.45	0	0.45	1	0.55
27	21	0.66	1	0.55	1	0.65	0	0.45	0	0.45
28	5	0.23	0	0.30	1	0.55	0	0.35	0	0.30
29	9	0.76	1	0.60	1	0.50	1	0.50	1	0.55
30	19	0.50	1	0.85	0	0.40	0	0.45	0	0.40
31	20	0.75	1	0.80	1	0.60	1	0.55	1	0.70
32	6	0.60	0	0.45	1	0.65	1	0.60	0	0.45
33	4	-0.04	0	0.25	1	0.55	0	0.30	0	0.30
34	29	0.37	1	0.60	0	0.40	0	0.35	0	0.40
35	6	0.11	0	0.35	0	0.35	0	0.35	0	0.35
36	9	0.95	1	0.70	1	0.65	1	0.80	1	0.75
37	24	0.84	1	0.65	1	0.90	1	0.70	1	0.70
38	9	0.95	1	0.70	1	0.65	1	0.65	1	0.60
39	16	0.35	1	0.70	1	0.55	0	0.40	1	0.50

40	6	0.60	1	0.55	1	0.65	0	0.45	0	0.45
41	22	0.67	1	0.55	1	0.60	1	0.55	0	0.45
42	4	-0.04	0	0.30	1	0.55	0	0.30	0	0.30
43	15	0.82	1	0.90	1	0.60	1	0.65	1	0.60
44	18	0.81	1	0.85	0	0.45	1	0.80	0	0.45
45	7	-0.00	0	0.40	0	0.45	0	0.35	0	0.40
46	16	0.62	1	0.80	0	0.45	1	0.60	0	0.45
47	16	0.62	1	0.70	0	0.45	1	0.55	1	0.50
48	24	0.53	1	0.75	1	0.70	1	0.75	1	0.75
49	26	0.44	1	0.80	1	0.55	1	0.80	1	0.75
50	18	0.79	1	0.85	1	0.85	1	0.85	1	0.85

Cuadro 5.1: Resultados comparativa puramente estructural.

		Algoritmo	Sujeto 1				Sujeto 2			
Operación	Nodos	Comparación	%E	%S	%T	Se Parecen	%E	%S	%T	Se Parecen
1	13	1.0	1.00	1.00	1.00	1	1.00	1.00	1.00	1
2	9	0.95	0.70	0.65	0.65	1	0.70	0.70	0.70	1
3	6	0.60	0.10	0.65	0.40	0	0.05	0.70	0.45	0
4	6	0.11	0.10	0.15	0.15	0	0.10	0.10	0.10	0
5	6	0.60	0.70	0.10	0.40	0	0.60	0.20	0.50	1
6	32	-0.39	0.10	0.60	0.40	0	0.15	0.55	0.45	0
7	31	0.60	0.55	0.65	0.50	1	0.60	0.65	0.60	1
8	8	0.88	0.70	0.30	0.50	1	0.65	0.20	0.50	1
9	17	0.86	0.65	0.80	0.70	1	0.70	0.75	0.75	1
10	15	0.91	0.65	0.60	0.60	1	0.60	0.60	0.60	1
11	16	0.62	0.50	0.65	0.55	1	0.45	0.70	0.60	1
12	4	-0.04	0.10	0.15	0.10	0	0.10	0.10	0.10	0
13	24	0.59	0.40	0.40	0.40	0	0.50	0.50	0.50	1
14	31	-0.36	0.10	0.40	0.30	0	0.20	0.40	0.45	0
15	6	0.60	0.10	0.70	0.40	0	0.20	0.65	0.45	0
16	10	0.77	0.50	1.00	0.80	1	0.60	1.00	0.70	1
17	18	0.78	0.35	0.65	0.50	1	0.40	0.60	0.45	0
18	14	0.72	0.60	0.65	0.60	1	0.60	0.60	0.60	1
19	6	0.60	0.10	0.65	0.40	0	0.10	0.55	0.40	0
20	5	0.36	0.10	0.35	0.25	0	0.10	0.40	0.30	0
21	34	-0.44	0.10	0.80	0.45	0	0.10	0.70	0.45	0
22	6	0.60	0.10	0.65	0.40	0	0.15	0.60	0.40	0
23	11	0.87	0.70	1.00	0.85	1	0.80	1.00	0.85	1
24	15	0.87	0.70	0.75	0.75	1	0.75	0.75	0.75	1
25	8	0.81	0.10	0.65	0.40	0	0.25	0.65	0.50	1

26	17	0.74	0.40	0.85	0.65	1	0.40	0.75	0.60	1
27	21	0.66	0.30	0.65	0.50	1	0.25	0.65	0.45	0
28	5	0.23	0.10	0.15	0.15	0	0.15	0.15	0.15	0
29	9	0.76	0.10	0.60	0.35	0	0.20	0.60	0.40	0
30	19	0.50	0.35	0.60	0.45	0	0.35	0.65	0.50	1
31	20	0.75	0.50	0.50	0.50	1	0.50	0.50	0.50	1
32	6	0.60	0.10	0.65	0.35	0	0.20	0.50	0.40	0
33	4	-0.04	0.10	0.15	0.15	0	0.15	0.20	0.25	0
34	29	0.37	0.45	0.45	0.45	0	0.45	0.45	0.45	0
35	6	0.11	0.10	0.15	0.15	0	0.15	0.15	0.15	0
36	9	0.95	0.50	0.65	0.60	1	0.60	0.65	0.60	1
37	24	0.84	0.50	0.70	0.60	1	0.50	0.70	0.60	1
38	9	0.95	0.65	0.65	0.65	1	0.60	0.60	0.60	1
39	16	0.35	0.65	0.10	0.40	0	0.70	0.20	0.50	1
40	6	0.60	0.65	0.10	0.40	0	0.70	0.20	0.50	0
41	22	0.67	0.30	0.40	0.35	0	0.25	0.40	0.25	0
42	4	-0.04	0.10	0.15	0.10	0	0.10	0.15	0.15	0
43	15	0.82	0.60	0.85	0.75	1	0.70	0.80	0.70	1
44	18	0.81	0.30	1.00	0.70	1	0.40	1.00	0.65	1
45	7	-0.00	0.10	0.15	0.15	0	0.15	0.15	0.15	0
46	16	0.62	0.30	0.65	0.45	0	0.40	0.70	0.55	1
47	16	0.62	0.30	0.65	0.45	0	0.35	0.70	0.50	0
48	24	0.53	0.70	0.40	0.55	1	0.60	0.30	0.45	0
49	26	0.44	0.65	0.55	0.60	1	0.70	0.60	0.60	1
50	18	0.79	0.70	0.65	0.70	1	0.70	0.60	0.65	1

Cuadro 5.2: Resultados comparativa estructural con información de nodos 1/2.

		Algoritmo	Sujeto 3				Sujeto 4			
Operación	Nodos	Comparación	%E	%S	%T	Se Parecen	%E	%S	%T	Se Parecen
1	13	1.0	1.00	1.00	1.00	1	1.00	1.00	1.00	1
2	9	0.95	0.65	0.60	0.60	1	0.65	0.65	0.65	1
3	6	0.60	0.15	0.60	0.40	0	0.15	0.55	0.50	1
4	6	0.11	0.15	0.10	0.10	0	0.15	0.20	0.20	0
5	6	0.60	0.55	0.15	0.40	0	0.70	0.20	0.55	1
6	32	-0.39	0.10	0.70	0.55	1	0.20	0.70	0.50	1
7	31	0.60	0.55	0.65	0.50	1	0.45	0.50	0.45	0
8	8	0.88	0.60	0.20	0.50	1	0.70	0.45	0.60	1
9	17	0.86	0.65	0.75	0.80	1	0.60	0.80	0.75	1
10	15	0.91	0.60	0.65	0.65	1	0.65	0.60	0.60	1
11	16	0.62	0.50	0.70	0.60	1	0.50	0.65	0.60	1

12	4	-0.04	0.05	0.10	0.10	0	0.15	0.15	0.15	0
13	24	0.59	0.45	0.35	0.45	0	0.40	0.45	0.40	0
14	31	-0.36	0.20	0.50	0.50	0	0.15	0.40	0.35	0
15	6	0.60	0.10	0.65	0.45	0	0.15	0.65	0.40	0
16	10	0.77	0.50	1.00	0.85	1	0.65	1.00	0.70	1
17	18	0.78	0.45	0.55	0.50	0	0.35	0.70	0.50	1
18	14	0.72	0.65	0.65	0.65	1	0.55	0.65	0.60	1
19	6	0.60	0.15	0.60	0.40	0	0.10	0.65	0.40	0
20	5	0.36	0.15	0.40	0.30	0	0.10	0.40	0.25	0
21	34	-0.44	0.10	0.70	0.40	0	0.10	0.75	0.40	0
22	6	0.60	0.10	0.60	0.35	0	0.10	0.65	0.40	0
23	11	0.87	0.80	1.00	0.80	1	0.75	1.00	0.80	1
24	15	0.87	0.75	0.75	0.75	1	0.70	0.80	0.75	1
25	8	0.81	0.15	0.60	0.45	0	0.10	0.60	0.40	0
26	17	0.74	0.45	0.80	0.65	1	0.40	0.85	0.65	1
27	21	0.66	0.30	0.65	0.50	1	0.40	0.50	0.35	0
28	5	0.23	0.10	0.15	0.10	0	0.10	0.20	0.10	0
29	9	0.76	0.15	0.65	0.40	0	0.10	0.55	0.35	0
30	19	0.50	0.30	0.65	0.45	0	0.35	0.65	0.50	1
31	20	0.75	0.50	0.50	0.50	1	0.50	0.50	0.50	1
32	6	0.60	0.10	0.55	0.30	0	0.15	0.60	0.35	0
33	4	-0.04	0.10	0.20	0.15	0	0.15	0.15	0.15	0
34	29	0.37	0.35	0.35	0.35	0	0.35	0.35	0.35	0
35	6	0.11	0.10	0.15	0.10	0	0.20	0.15	0.15	0
36	9	0.95	0.45	0.70	0.55	1	0.45	0.65	0.55	1
37	24	0.84	0.55	0.65	0.60	1	0.45	0.70	0.55	1
38	9	0.95	0.65	0.65	0.65	1	0.65	0.65	0.65	1
39	16	0.35	0.60	0.10	0.35	0	0.65	0.15	0.45	0
40	6	0.60	0.55	0.10	0.30	0	0.60	0.10	0.40	0
41	22	0.67	0.25	0.45	0.35	0	0.30	0.45	0.45	0
42	4	-0.04	0.10	0.10	0.10	0	0.20	0.15	0.15	0
43	15	0.82	0.60	0.80	0.70	1	0.65	0.85	0.75	1
44	18	0.81	0.20	1.00	0.65	1	0.30	1.00	0.70	1
45	7	-0.00	0.10	0.20	0.15	0	0.10	0.10	0.10	0
46	16	0.62	0.30	0.60	0.35	0	0.35	0.70	0.50	1
47	16	0.62	0.25	0.55	0.40	0	0.35	0.70	0.50	1
48	24	0.53	0.75	0.45	0.60	1	0.65	0.45	0.55	1
49	26	0.44	0.60	0.55	0.55	1	0.65	0.55	0.60	1
50	18	0.79	0.65	0.55	0.55	1	0.70	0.65	0.70	1

Cuadro 5.3: Resultados comparativa estructural con información de nodos 2/2.

La comparación del conjunto de pruebas provista por el algoritmo se midió comparando los resultados obtenidos por los voluntarios, con la finalidad de obtener un umbral U que especifique el porcentaje a partir del cual una comparación puede ser confiable.

5.4.1. Resultados para comparaciones estructurales

Los resultados de esta sección se calcularon en base a la información provista por el cuadro 5.1.

5.4.1.1. Resultados para un umbral de 65 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 6, 8, 9, 10, 11, 13, 14, 16, 17, 18, 21, 23, 24, 26, 27, 29, 30, 31, 34, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 18, 23, 24, 26, 27, 29, 31, 36, 37, 38, 41, 43, 44, 50\}$

A: 21 **B:** 13 **C:** 1

Precision: 95.45454545454545

Recall: 61.76470588235294

Sujeto 2

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 31, 32, 33, 36, 37, 38, 39, 40, 41, 42, 43, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 18, 23, 24, 25, 27, 29, 31, 36, 37, 38, 41, 43, 50\}$

A: 20 **B:** 14 **C:** 2

Precision: 90.9090909090909

Recall: 58.82352941176471

Sujeto 3

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 14, 16, 17, 18, 22, 23, 24, 25, 29, 31, 32, 36, 37, 38, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 18, 23, 24, 25, 29, 31, 36, 37, 38, 41, 43, 44, 50\}$

A: 20 **B:** 10 **C:** 2

Precision: 90.9090909090909

Recall: 66.66666666666666

Sujeto 4

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 18, 19, 23, 24, 25, 26, 29, 31, 36, 37, 38, 39, 43, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 18, 23, 24, 25, 26, 29, 31, 36, 37, 38, 43, 50\}$

A: 18 **B:** 8 **C:** 4

Precision: 81.81818181818183

Recall: 69.23076923076923

Se puede estimar un porcentaje de "precision" de 89,76 % y un porcentaje de "recall" de 64,11 % para un umbral de 65 %.

5.4.1.2. Resultados para un umbral de 75 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 6, 8, 9, 10, 11, 13, 14, 16, 17, 18, 21, 23, 24, 26, 27, 29, 30, 31, 34, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 23, 24, 29, 31, 36, 37, 38, 43, 44, 50\}$

A: 17 **B:** 17 **C:** 1

Precision: 94.44444444444444

Recall: 50.0

Sujeto 2

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 31, 32, 33, 36, 37, 38, 39, 40, 41, 42, 43, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 23, 24, 25, 29, 31, 36, 37, 38, 43, 50\}$

A: 17 **B:** 17 **C:** 1

Precision: 94.44444444444444

Recall: 50.0

Sujeto 3

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 14, 16, 17, 18, 22, 23, 24, 25, 29, 31, 32, 36, 37, 38, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 23, 24, 25, 29, 31, 36, 37, 38, 43, 44, 50\}$

A: 18 **B:** 12 **C:** 0

Precision: 100.0

Recall: 60.0

Sujeto 4

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 18, 19, 23, 24, 25, 26, 29, 31, 36, 37, 38, 39, 43, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 23, 24, 25, 29, 31, 36, 37, 38, 43, 50\}$

A: 16 **B:** 10 **C:** 2

Precision: 88.88888888888889

Recall: 61.53846153846154

Se puede estimar un porcentaje de "precision" de 94,44 % y un porcentaje de "recall" de 55,38 % para un umbral de 75 %.

5.4.1.3. Resultados para un umbral de 85 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 6, 8, 9, 10, 11, 13, 14, 16, 17, 18, 21, 23, 24, 26, 27, 29, 30, 31, 34, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 25 **C:** 0

Precision: 100.0

Recall: 26.47058823529412

Sujeto 2

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27, 28, 29, 31, 32, 33, 36, 37, 38, 39, 40, 41, 42, 43, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 25 **C:** 0

Precision: 100.0

Recall: 26.47058823529412

Sujeto 3

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 14, 16, 17, 18, 22, 23, 24, 25, 29, 31, 32, 36, 37, 38, 41, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 21 **C:** 0

Precision: 100.0

Recall: 30.0

Sujeto 4

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 13, 16, 18, 19, 23, 24, 25, 26, 29, 31, 36, 37, 38, 39, 43, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 17 **C:** 0

Precision: 100.0

Recall: 34.61538461538461

Se puede estimar un porcentaje de "precision" de 100 % y un porcentaje de "recall" de 29,38 % para un umbral de 80 %.

5.4.2. Resultados para comparaciones estructurales con tipos de nodos

Los resultados de esta sección se calcularon en base a la información provista por los cuadros 5.2, 5.3.

5.4.2.1. Resultados para un umbral de 65 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 50\}$

A: 19 **B:** 4 **C:** 3

Precision: 86.36363636363636

Recall: 82.6086956521739

Sujeto 2

Operaciones relevantes:

$\{1, 2, 5, 7, 8, 9, 10, 11, 13, 16, 18, 23, 24, 25, 26, 30, 31, 36, 37, 38, 39, 43, 44, 46, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 18, 23, 24, 25, 26, 31, 36, 37, 38, 43, 44, 50\}$

A: 18 **B:** 8 **C:** 4

Precision: 81.81818181818183

Recall: 69.23076923076923

Sujeto 3

Operaciones relevantes:

$\{1, 2, 6, 7, 8, 9, 10, 11, 16, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 50\}$

A: 18 **B:** 5 **C:** 4

Precision: 81.81818181818183

Recall: 78.26086956521739

Sujeto 4

Operaciones relevantes:

$\{1, 2, 3, 5, 6, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 30, 31, 36, 37, 38, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 18, 23, 24, 26, 31, 36, 37, 38, 43, 44, 50\}$

A: 18 **B:** 9 **C:** 4

Precision: 81.81818181818183

Recall: 66.66666666666666

Se puede estimar un porcentaje de "precision" de 82,94 % y un porcentaje de "recall" de 74,18 % para un umbral de 65 %.

5.4.2.2. Resultados para un umbral de 75 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 23, 24, 31, 36, 37, 38, 43, 44, 50\}$

A: 16 **B:** 7 **C:** 2

Precision: 88.88888888888889

Recall: 69.56521739130434

Sujeto 2

Operaciones relevantes:

$\{1, 2, 5, 7, 8, 9, 10, 11, 13, 16, 18, 23, 24, 25, 26, 30, 31, 36, 37, 38, 39, 43, 44, 46, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 23, 24, 25, 31, 36, 37, 38, 43, 44, 50\}$

A: 16 **B:** 10 **C:** 2

Precision: 88.88888888888889

Recall: 61.53846153846154

Sujeto 3

Operaciones relevantes:

$\{1, 2, 6, 7, 8, 9, 10, 11, 16, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 23, 24, 31, 36, 37, 38, 43, 44, 50\}$

A: 15 **B:** 8 **C:** 3

Precision: 83.33333333333334

Recall: 65.21739130434783

Sujeto 4

Operaciones relevantes:

$\{1, 2, 3, 5, 6, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 30, 31, 36, 37, 38, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 16, 17, 23, 24, 31, 36, 37, 38, 43, 44, 50\}$

A: 16 **B:** 11 **C:** 2

Precision: 88.88888888888889

Recall: 59.25925925925925

Se puede estimar un porcentaje de "precision" de 87,49 % y un porcentaje de "recall" de 63,88 % para un umbral de 75 %.

5.4.2.3. Resultados para un umbral de 85 %

Sujeto 1

Operaciones relevantes:

$\{1, 2, 7, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes Recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 14 **C:** 0

Precision: 100.0

Recall: 39.130434782608695

Sujeto 2

Operaciones relevantes:

$\{1, 2, 5, 7, 8, 9, 10, 11, 13, 16, 18, 23, 24, 25, 26, 30, 31, 36, 37, 38, 39, 43, 44, 46, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 17 **C:** 0

Precision: 100.0

Recall: 34.61538461538461

Sujeto 3

Operaciones relevantes:

$\{1, 2, 6, 7, 8, 9, 10, 11, 16, 18, 23, 24, 26, 27, 31, 36, 37, 38, 43, 44, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 14 **C:** 0

Precision: 100.0

Recall: 39.130434782608695

Sujeto 4

Operaciones relevantes:

$\{1, 2, 3, 5, 6, 8, 9, 10, 11, 16, 17, 18, 23, 24, 26, 30, 31, 36, 37, 38, 43, 44, 46, 47, 48, 49, 50\}$

Operaciones relevantes recuperadas:

$\{1, 2, 8, 9, 10, 23, 24, 36, 38\}$

A: 9 **B:** 18 **C:** 0

Precision: 100.0

Recall: 33.33333333333333

Se puede estimar un porcentaje de "precision" de 100 % y un porcentaje de "recall" de 36,55 % para un umbral de 80 %.

5.4.3. Interpretación de resultados

Se realizaron dos tipos de comparaciones humanas, con la finalidad de determinar la efectividad del algoritmo en determinar la similitud entre operaciones de servicios Web. La primer comparación humana consistió en establecer la similitud entre los árboles de operaciones de forma puramente estructural, ignorando el contenido de los nodos (figura 5.5) en cada árbol comparado. La segunda comparación humana tomó en cuenta el contenido de los nodos. Realizar dos tipos de comparaciones humanas para contrastarlas con el mismo resultado emitido por el algoritmo de comparación estructural, permitieron observar si los criterios de ordenamiento y el orden de construcción de los vectores característicos que se propuso es adecuado para representar los elementos de operaciones de servicios Web. Mostrando que el tipo de ordenamiento implementado en el algoritmo permite obtener comparaciones con una precisión aceptable, la cual posee una ligera variación cuando se toma en cuenta el contenido de los nodos de un árbol y cuando no y demuestra que el algoritmo puede ser utilizado con un buen grado de confiabilidad para comparar cualquier estructura de árbol independientemente de su contexto.

Se calcularon las mediciones "precision" and "recall" para los resultados emitidos por las comparaciones humanas, variando el umbral de aceptación que supone a los elementos de una comparación tener un alto grado de similitud. Se utilizaron umbrales de 65 % ,75 % y 85 % tanto para la comparación puramente estructural como la comparación estructural con información sobre el tipo de nodos.

Se utilizó el mismo porcentaje de correlación emitido por el algoritmo para contrastar los porcentajes de similitud de los dos tipos de comparaciones humanas, demostrando que para ambos casos las correlaciones pueden ser confiables para umbrales iguales o superiores al 65 %.

Las figuras 5.5 y 5.5 muestran las gráficas de los valores calculados de "precision" y "recall" para los dos experimentos realizados, y permiten apreciar en ambos experimentos que los valores de "precision" incrementan en medida que se incrementa el umbral, sucediendo de manera inversa en el caso del "recall", de manera que; a umbrales más altos, mayor precisión del algoritmo y menor capacidad en la detección de similitudes más débiles.

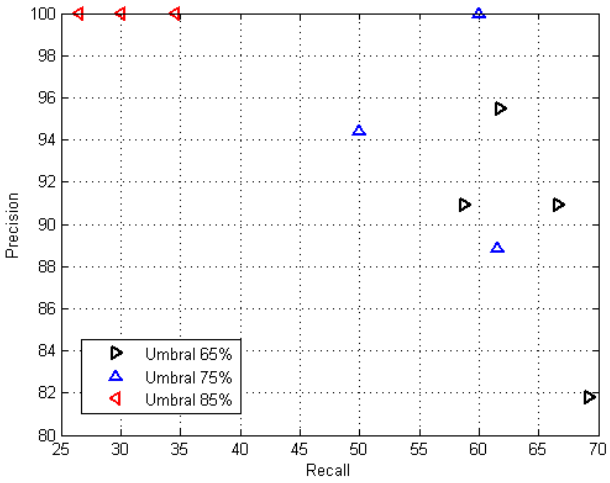


Figura 5.4: Gráfica para comparaciones estructurales.

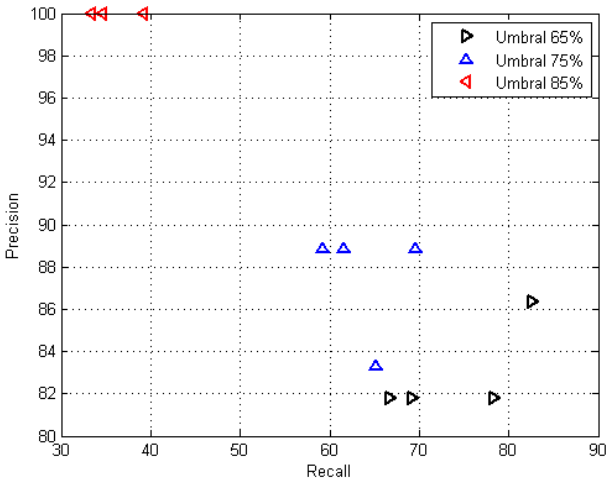


Figura 5.5: Gráfica para comparaciones estructurales con tipos de nodos.

Se puede concluir que a partir de umbrales mayores o iguales 85 % el grado de precisión del algoritmo es muy alto. Sin embargo, su capacidad para encontrar posibles operaciones compatibles es casi nula. Por otro lado, con un umbral mayor o igual a 65 % y menor a 75 % ofrece una precisión muy aceptable en la comparación de operaciones y aun es capaz de detectar posibles operaciones compatibles.

Finalmente, en base a los resultados obtenidos, se puede decir que el algoritmo desarrollado puede ser utilizado en la estimación de similitudes entre operaciones de servicios Web con un alto grado de confiabilidad. Además, se puede decir que el algoritmo puede ser reimplantado en contextos diferentes que hagan uso de comparaciones estructurales entre arboles N-arios.

5.5. Sitio Web

El sitio proporciona cuatro controles principales sobre su interfaz de usuario, con los cuales se despliegan las funcionalidades y se muestran los resultados.

La figura 5.6 muestra los principales controles en la interfaz del sitio Web, los cuales se encuentran numerados y son descritos a continuación:

1. Posibles acciones de un usuario en el sitio Web:

   -Comparación entre las operaciones de dos archivos WSDL.

   -Comparación entre las operaciones de un archivo WSDL y las de todo un repositorio WSDL.

   -Comparación entre todas las operaciones de un repositorio WSDL.

   -Subir especificaciones WSDL y realizar una comparación entre todas las operaciones encontradas.

   -Subir especificaciones WSDL y descargar en formato ZIP los archivos RDF pertenecientes a las operaciones contenidas en los archivos previamente subidos.

2. Muestra los resultados de las comparaciones realizadas mediante los siguientes campos:

   -WS A: Nombre del primer servicio Web a comparar.

   -Operación A: Nombre de la operación a comparada perteneciente al primer servicio Web.

   -WS B: Nombre del segundo servicio Web a comparar.

   -Operación B: Nombre de la operación a comparada perteneciente al segundo servicio Web.

   - %Operation: Porcentaje de correlación estructural para las dos operaciones.

   - %Input: Porcentaje de correlación estructural entre los mensajes de entrada de ambas operaciones.

   - %Output: Porcentaje de correlación estructural entre los mensajes de salida de ambas operaciones.

3. Permite almacenar la información contenida en 2 en formatos PDF, CSV, XLS y XML

4. Muestra a detalle la información de una fila en 2

La figura 5.7 muestra el detalle de una selección sobre la tabla de resultados de la figura 5.8, y se compone de los siguientes elementos:

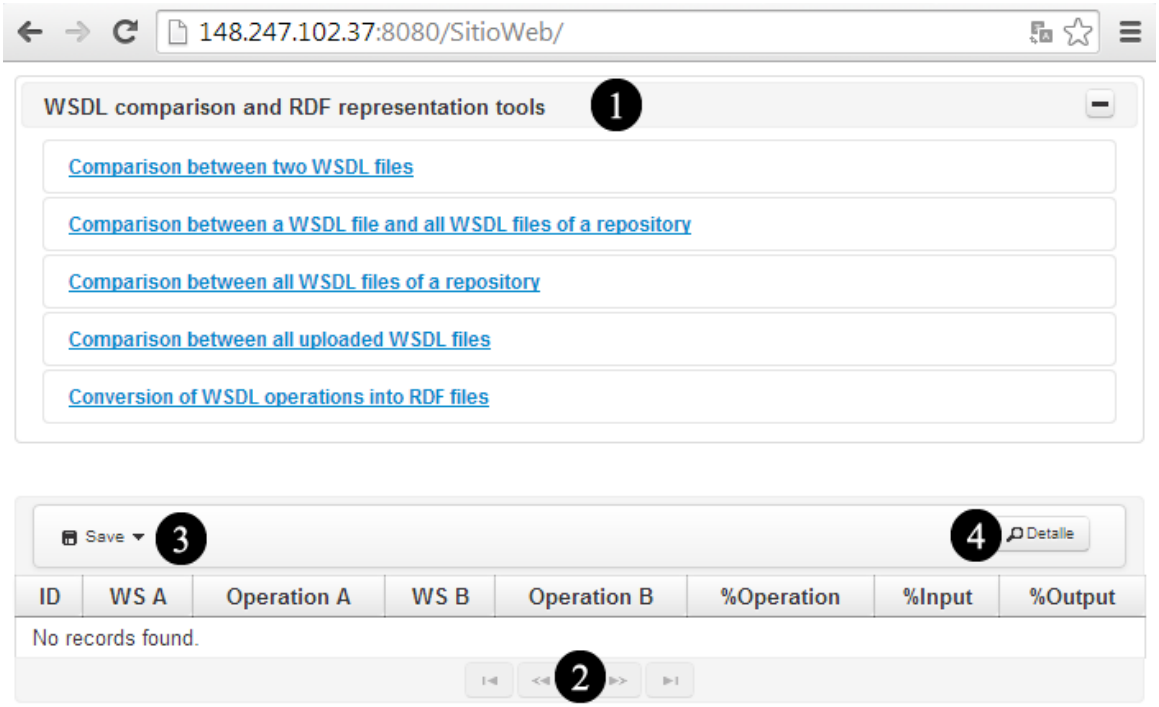


Figura 5.6: Interfaz de usuario: ventana principal.

1. Muestra el resultado de la correlación entre las operaciones, los mensajes de entrada y los mensajes de salida de las operaciones.

2. Muestra el grafo de las operaciones.

3. Indica los nombres de las operaciones.

4. Proporciona una cadena de texto en el lenguaje de descripción de grafos DOT para que los grafos de las operaciones puedan ser fácilmente reproducidos.

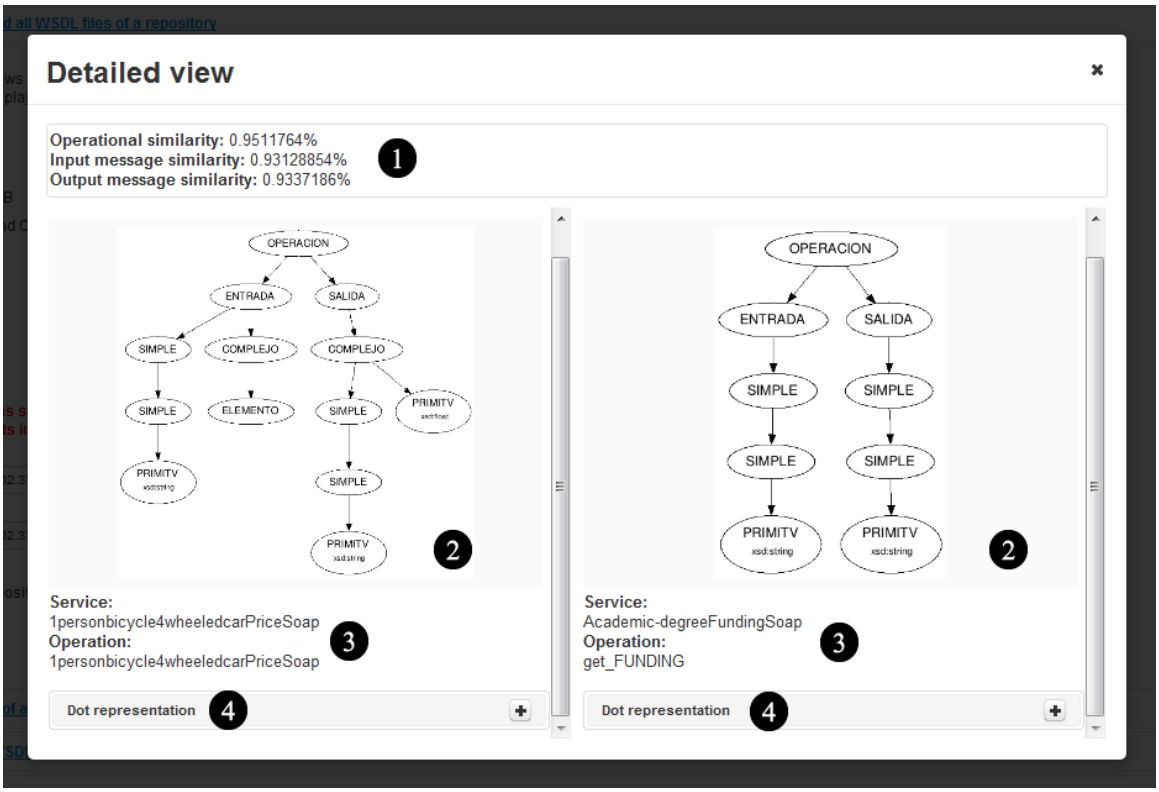
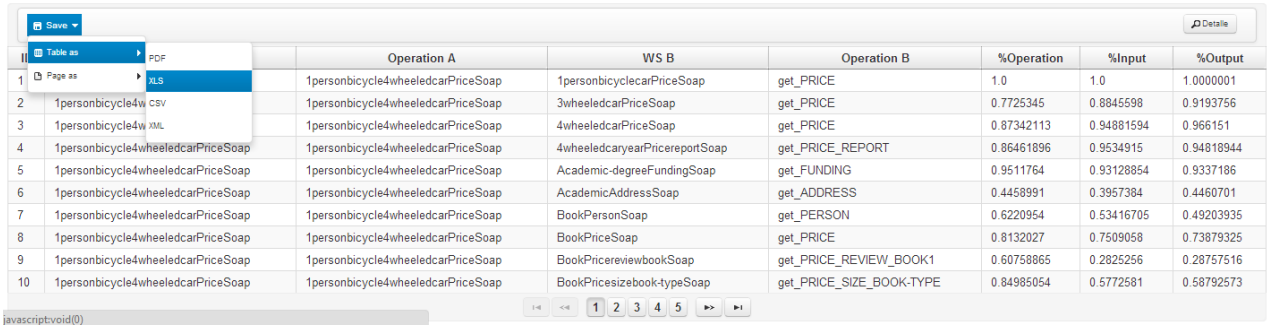


Figura 5.7: Interfaz de usuario: ventana de detalle para comparaciones seleccionadas.



		Operation A	WS B	Operation B	%Operation	%Input	%Output
1	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	1personbicyclecarPriceSoap	get_PRICE	1.0	1.0	1.0000001
2	1personbicycle4w	1personbicycle4wheeledcarPriceSoap	3wheeledcarPriceSoap	get_PRICE	0.7725345	0.8845598	0.9193756
3	1personbicycle4w	1personbicycle4wheeledcarPriceSoap	4wheeledcarPriceSoap	get_PRICE	0.87342113	0.94881594	0.966151
4	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	4wheeledcaryearPricereportSoap	get_PRICE_REPORT	0.86461896	0.9534915	0.94818944
5	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	Academic-degreeFundingSoap	get_FUNDING	0.9511764	0.93128854	0.9337186
6	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	AcademicAddressSoap	get_ADDRESS	0.4458991	0.3957384	0.4460701
7	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	BookPersonSoap	get_PERSON	0.6220954	0.53416705	0.49203935
8	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	BookPriceSoap	get_PRICE	0.8132027	0.7509058	0.73879325
9	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	BookPricereviewbookSoap	get_PRICE_REVIEW_BOOK1	0.60758865	0.2825256	0.28757516
10	1personbicycle4wheeledcarPriceSoap	1personbicycle4wheeledcarPriceSoap	BookPricesizebook-typeSoap	get_PRICE_SIZE_BOOK-TYPE	0.84985054	0.5772581	0.58792573

1 2 3 4 5

Figura 5.8: Interfaz de usuario: tabla de resultados.

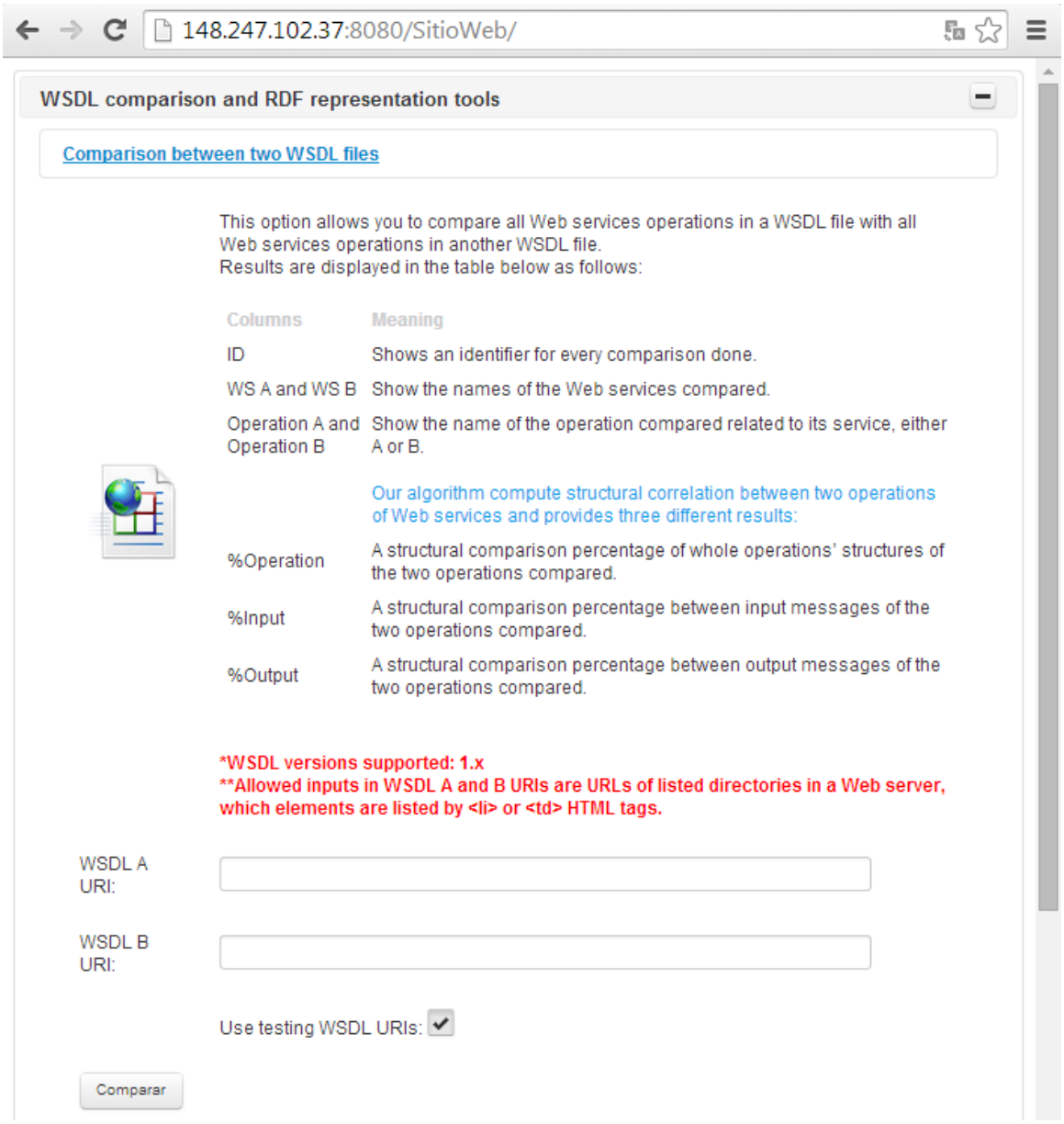


Figura 5.9: Interfaz de usuario: comparación entre dos especificaciones WSDL.

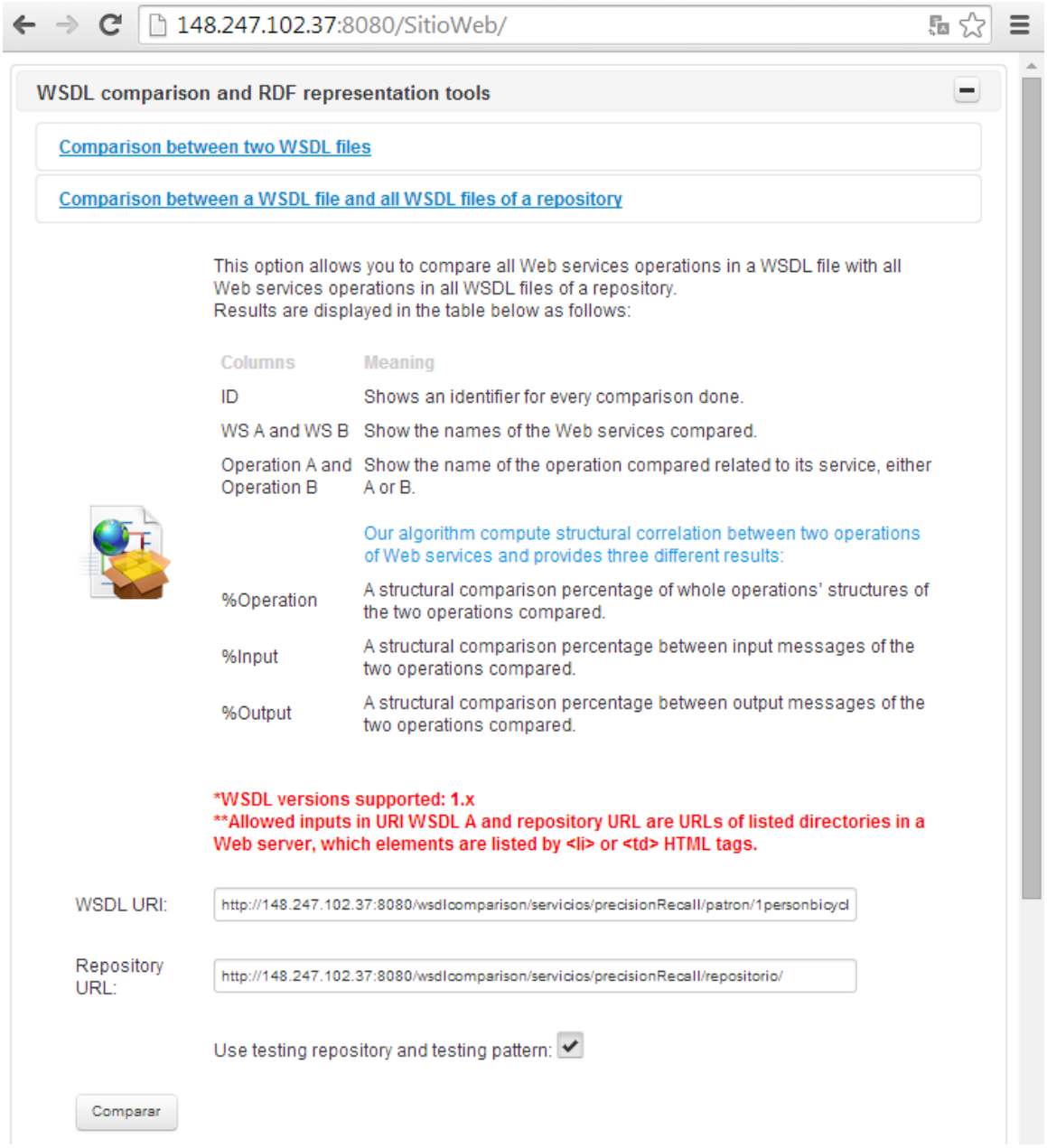


Figura 5.10: Interfaz de usuario: comparación entre una especificaciones WSDL y un repositorio WSDL.

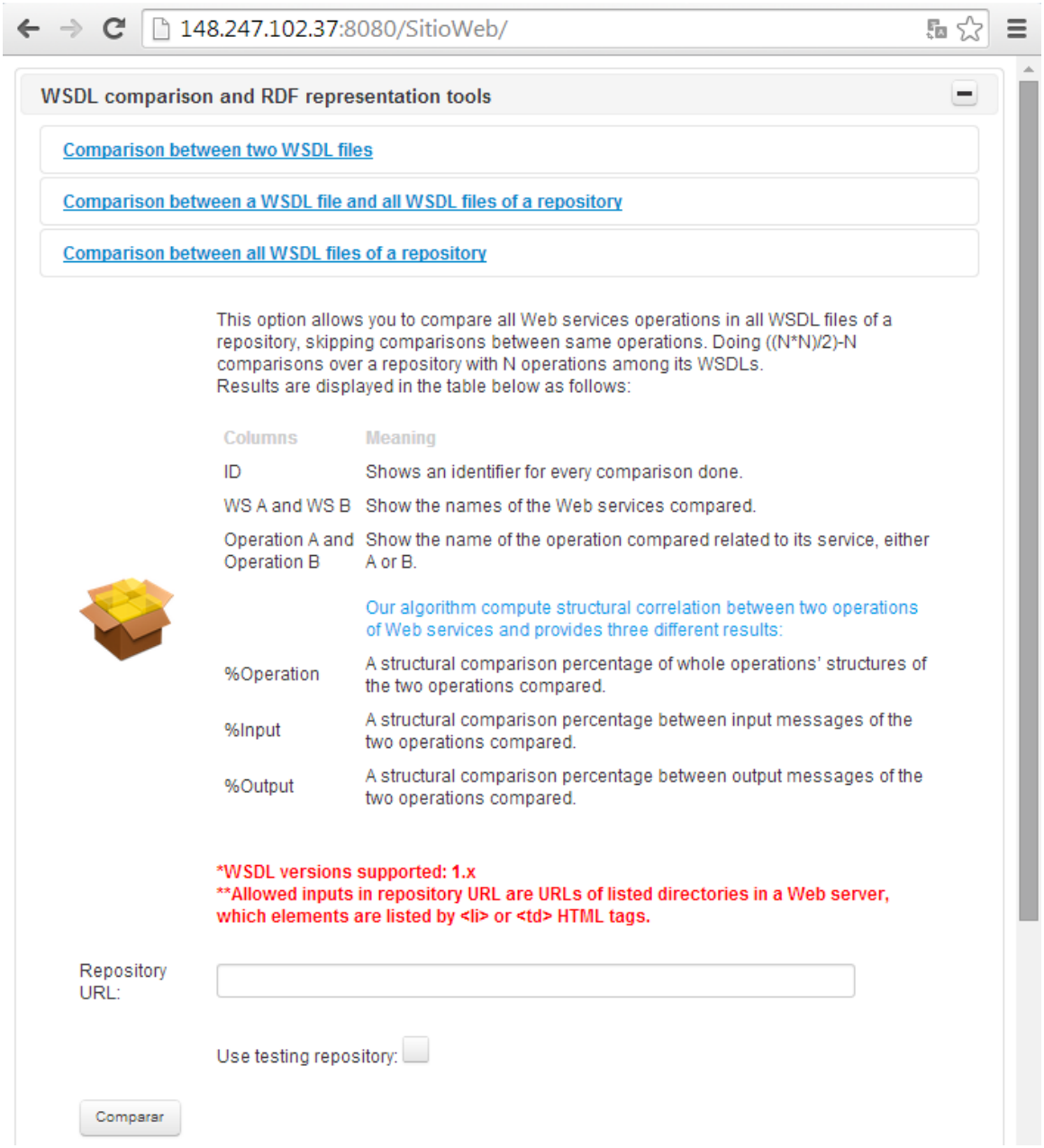


Figura 5.11: Interfaz de usuario: comparación de un repositorio WSDL.

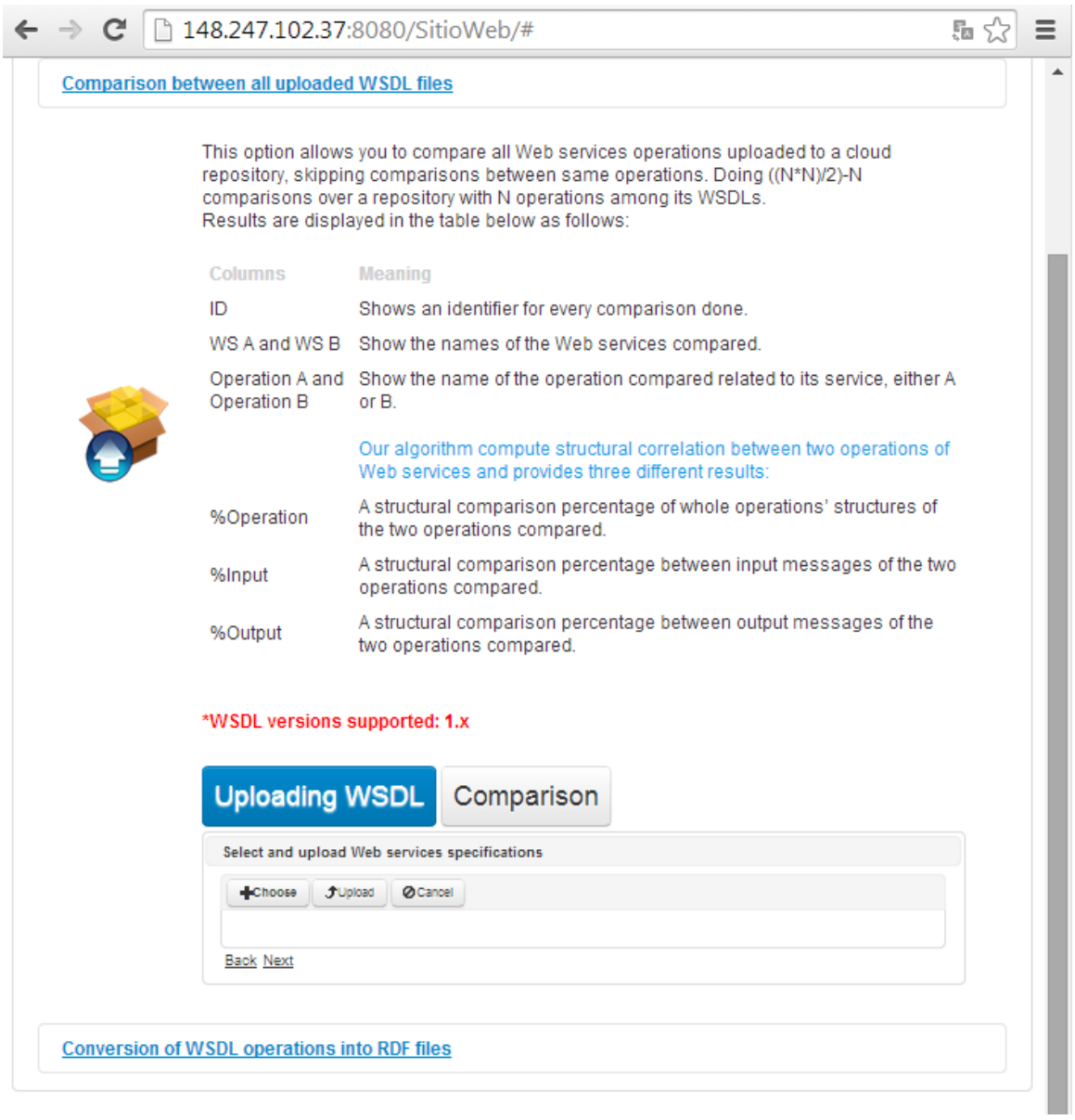


Figura 5.12: Interfaz de usuario: comparación de un repositorio de especificaciones WSDL cargadas al servidor.

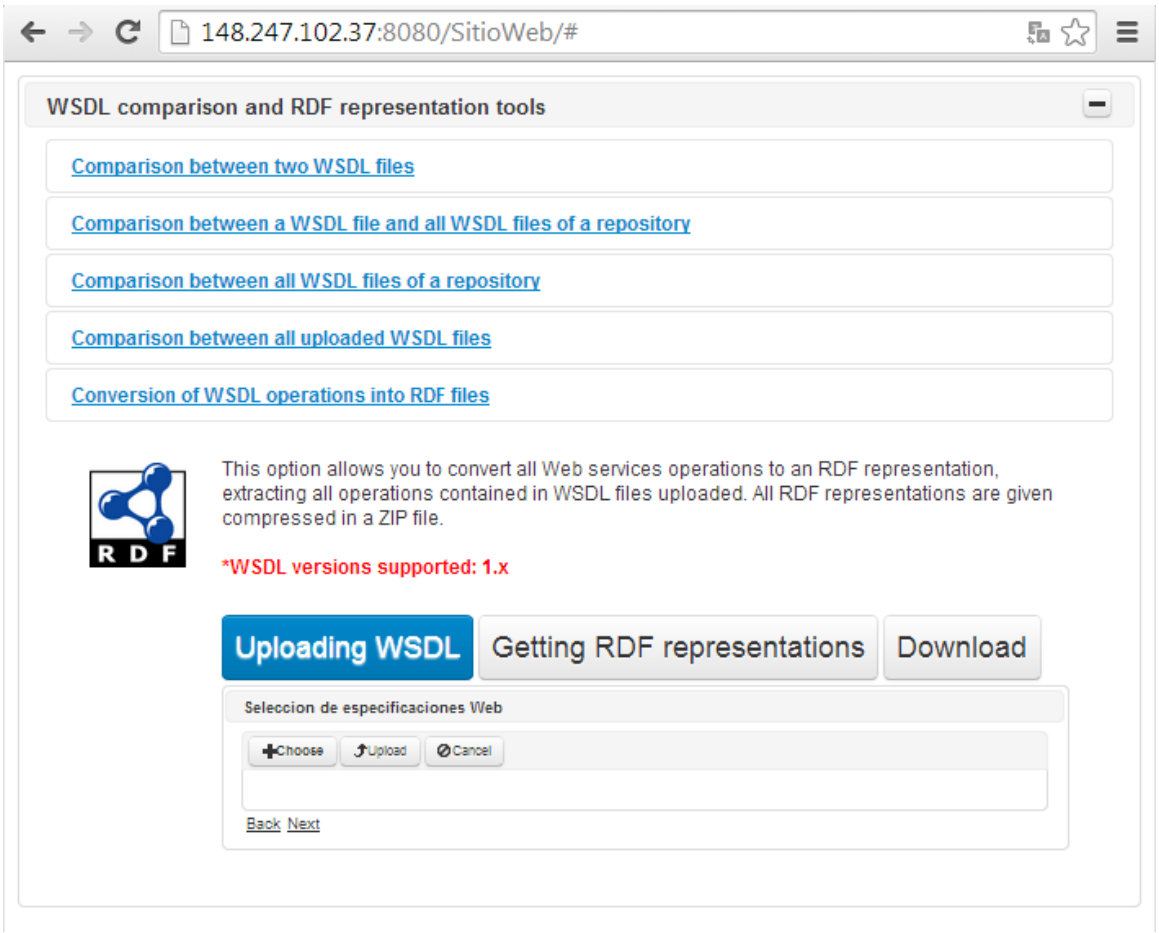


Figura 5.13: Interfaz de usuario: conversor de operaciones WSDL a formato RDF.

Figura 5.14: Interfaz de usuario: descarga de operaciones convertidas a formato RDF.

Capítulo 6

Conclusiones

Existen diferentes alternativas para la comparación de operaciones de servicios Web, mismas que abordan la problemática con la finalidad de mejorar la clasificación y composición de servicios, la construcción de redes de recomendación de servicios o simplemente la de otorgar herramientas de desarrollo para los usuarios que implementan servicios Web. La comparación de servicios Web es una tarea complicada, debido a que involucra diversos criterios para determinar similitudes entre operaciones. Por lo anterior, no existen "Benchmarks" para cada enfoque de comparación y hasta el momento solo es posible tratar de obtener el rendimiento y la efectividad de los comparadores de operaciones de servicios Web mediante métodos que permiten contraponer los resultados de una comparativa contra los de un patrón avalado por expertos de la materia.

Las pruebas realizadas en el algoritmo desarrollado revelaron la efectividad del mismo para realizar estimaciones de similitud entre servicios Web. Mostrando resultados confiables que asemejaron en gran medida a los patrones humanamente medidos. Arrojó valores aceptables que avalan su confiabilidad para ser ampliamente utilizado por aplicaciones de clasificación, comparación y composición de servicios Web.

Existen tendencias que intentan comparar operaciones de servicios Web para construir redes de recomendación entre servicios, tomando en cuenta únicamente a la semántica involucrada en cada operación, y así catalogar temáticamente a cada servicio. Por otro lado, comparar operaciones por su estructura otorgaría una dimensión más a las clasificaciones de operaciones, ya que proporciona información acerca de cómo se encuentran definidos los tipos de dato de una operación, facilitando el reconocimiento de los parámetros de invocación para cada operación. Es por esto, que la integración de algoritmos de comparación estructural como el que esta tesis propone en sistemas clasificadores para la construcción redes de recomendación de servicios Web, ayudarían a realizar recomendaciones más robustas, manteniendo relaciones entre servicios cuya similitud no sea únicamente el aspecto

semántico sino también la estructura de tipos que permiten sus invocaciones.

Gran parte de los enfoques de la comparación de operaciones de SW utilizan modelos de representación de datos y algoritmos incompatibles entre sí. Proponer un modelo de representación básico para operaciones de SW que no sea ajeno a la estructura jerárquica natural de un documento WSDL, proporcionará un escenario común para la integración de diversos algoritmos sin propiciar afectaciones mutuas. En consecuencia, representar operaciones en estructuras de árbol N-arias facilitará el manejo de las mismas dentro de cualquier algoritmo de comparación estructural o semántica, haciendo compatibles a dichos algoritmos por el uso base de esta representación.

Plantear el uso de serializaciones RDF como versiones semánticas de las operaciones de un servicio Web, permitirá reutilizar gran parte de los algoritmos de comparación semántica existentes. Debido a que RDF proporciona una versión semántica de la información, la cual es reutilizable y legible por sistemas de cómputo, permitiendo enlazar los nombres de las operaciones de SW con descripciones semánticas asociadas descritas en repositorios de conocimiento.

Implementar todos los elementos de la comparación de operaciones en una API, otorgará la facilidad de utilizar individualmente cada módulo involucrado. Los módulos de análisis y extracción de operaciones mediante archivos de especificación WSDL y el módulo de representación de operaciones en estructuras de árbol N-arias, podrán ser fácilmente utilizados por cualquier aplicación Java de escritorio o Web.

El sitio Web desarrollado proporcionará una herramienta de decisión para los interesados en conocer la similitud estructural existente entre las operaciones de diferentes servicios Web, proporcionando diversas opciones de comparación y entregando a detalle la información calculada en cada comparación. Además, la aplicación Web permite a los usuarios obtener representaciones semánticas en formato RDF, construidas a partir del modelo de representación semántica que propone esta tesis.

6.1. Trabajo a futuro

Extraer la definición un servicio Web es una tarea sencilla, ya que el lenguaje WSDL posee un formato XML de etiquetas predefinidas que permiten analizar concretamente una especificación de un servicio Web. Sin embargo, extraer la información de los tipos de dato que componen a las operaciones de los servicios Web representa una tarea complicada, debido a que XML Schema, el lenguaje de definición de tipos utilizado por WSDL, ofrece una gran variedad expresiva para representar tipos de dato. La expresividad de XML Schema permite definir un mismo tipo de dato de diferentes maneras, dependiendo el propósito que tenga cada definición. Por lo anterior, analizar automáticamente la estructura de tipos de dato de una especificación de servicio Web representa una

complicación, por lo cual se propone como trabajo a futuro robustecer el vocabulario del módulo de análisis y extracción de operaciones a fin de soportar la lectura más expresiones para la definición de tipos de dato.

Expresar los elementos de una definición de tipos en los nodos de un árbol N-ario proporciona una visión general suficiente para comprender la estructura de una operación de un servicio Web. Sin embargo, la definición de tipos de XML Schema a través de los indicadores "Sequence", "Choice", "All", "maxOccurs" y "MinOcurrs", permite especificar la cantidad de apariciones de una misma variable, así como su orden de aparición, lo cual otorgaría mayor información acerca de una operación de servicio Web. Por tanto, proponemos la creación de un formato complementario a las estructuras de árbol N-arias, que permita expresar la información de los indicadores de XML Schema y pueda ser comparables por el algoritmo propuesto.

El algoritmo propuesto es un procedimiento estandard para comparar estructuras de árbol N-arias y puede ser utilizado en cualquier contexto que requiera la comparación de árboles. Sin embargo, el desempeño del algoritmo sólo se ha comprobado para estructuras pequeñas, propias de las operaciones promedio de los servicios Web. Se propone como trabajo a futuro implementar el algoritmo de comparación a diferentes contextos que lo obliguen a ejecutarse en alto rendimiento, ya que este algoritmo propone un método estadístico que podría competir ampliamente con los tiempos de algoritmos que realizan comparaciones mediante exhaustividad y combinatoria.

Se propone la reutilización de los vocabularios existentes en repositorios de información (e.g. Dbpedia) para realizar enriquecer automáticamente las representaciones RDF propuestas, generando tripletas cuyos sujetos sean las variables de las operaciones y cuyos objetos sean los recursos semánticos del repositorio. De manera que la documentación de un servicio se realice de automáticamente tras asociar el nombre de cada variable de operación con toda la información asociada contenida en diferentes repositorios de conocimiento.

Apéndice A

Conjunto de Pruebas

La siguiente tabla muestra el conjunto de servicios Web utilizado para efectuar las mediciones sobre el algoritmo de comparación estructural propuesto.

No	**Operación**	**Nodos**
P	WSDL: 1personbicycle4wheeledcar_price_service.wsdl Servicio: 1personbicycle4wheeledcarPriceSoap Operación: get_PRICE	13
1	WSDL: 1personbicyclecar_price_TheBestservice.wsdl Servicio: 1personbicyclecarPriceSoap Operación: get_PRICE	13
2	WSDL: Academic-degreeFunding.wsdl Servicio: Academic-degreeFundingSoap Operación: get_FUNDING	9
3	WSDL: _weatherfront_Germanyservice.wsdl Servicio: WeatherfrontSoap Operación: get_WEATHERFRONT	6
4	WSDL: _coffeeteareport_service.wsdl Servicio: CoffeeteareportSoap Operación: get_COFFEE_TEA_REPORT	6
5	WSDL: TitleSaving_service.wsdl Servicio: TitleSoap Operación: get	6

6	WSDL: Cameraprice.wsdl Servicio: CamerapriceSoap Operación: get_CAMERA_PRICE	32
7	WSDL: BookPricereviewbook.wsdl Servicio: BookPricereviewbookSoap Operación: get_PRICE_REVIEW_BOOK1	31
8	WSDL: TitleFilm.wsdl Servicio: TitleFilmSoap Operación: get_FILM	8
9	WSDL: 4wheeledcaryear_pricereport_service.wsdl Servicio: 4wheeledcaryearPricereportSoap Operación: get_PRICE_REPORT	17
10	WSDL: Booknonmedicaltransport.wsdl Servicio: BooknonmedicaltransportSoap Operación: getBookNonMedicalTransport_SeatNumberBookNonMedicalTransport_BookingNumber	15
11	WSDL: BookRecommendedprice.wsdl Servicio: BookRecommendedpriceSoap Operación: get_RECOMMENDEDPRICE	16
12	WSDL: _lecturer-in-academiaMunchenUniversity_service.wsdl Servicio: Lecturer-in-academiaSoap Operación: get_LECTURER-IN-ACADEMIA	4
13	WSDL: gazetteerLookupLocation.wsdl Servicio: GazetteerLocationSoap Operación: get_MATCHED-LOCATION_MAP_LATITUDE_LONGITUDE	24
14	WSDL: Camerataxedprice.wsdl Servicio: CamerataxedpriceSoap Operación: get_CAMERA_TAXEDPRICE	31
15	WSDL: Food.wsdl Servicio: FoodSoap Operación: get_FOOD	6
16	WSDL: 3wheeledcar_price_service.wsdl Servicio: 3wheeledcarPriceSoap Operación: get_PRICE	10

17	WSDL: calculateSunriseTime.wsdl Servicio: CalculateSunriseSoap Operación: get_SUNRISE	18
18	WSDL: university_researcher_service.wsdl Servicio: UniversityResearcherSoap Operación: get_RESEARCHER	14
19	WSDL: _videomediaSaturn_service.wsdl Servicio: VideomediaSoap Operación: get_VIDEOMEDIA	6
20	WSDL: Film.wsdl Servicio: FilmSoap Operación: get_FILM	5
21	WSDL: Camerataxedpricedutytax.wsdl Servicio: CamerataxedpricedutytaxSoap Operación: get_CAMERA_TAXEDPRICE_DUTYTAX	34
22	WSDL: _warmfront_Italyservice.wsdl Servicio: WarmfrontSoap Operación: get_WARMFRONT	6
23	WSDL: 4wheeledcar_price_service.wsdl Servicio: 4wheeledcarPriceSoap Operación: get_PRICE	11
24	WSDL: car_taxedpriceprice_service.wsdl Servicio: CarTaxedpricepriceSoap Operación: get_TAXEDPRICE_PRICE	15
25	WSDL: Filmvideomedia.wsdl Servicio: FilmvideomediaSoap Operación: get_FILM_VIDEOMEDIA	8
26	WSDL: BookTaxedprice1.wsdl Servicio: BookTaxedpriceSoap Operación: get_TAXEDPRICE	17
27	WSDL: findNearbyPostalCodes.wsdl Servicio: NearbyPostalCodesSoap Operación: get_POSTALCODE	21

28	WSDL: _droughtreport_service.wsdl Servicio: DroughtreportSoap Operación: get_DROUGHT_REPORT	5
29	WSDL: UnsuccessfulDiagnosis_service.wsdl Servicio: DiagnosticprocessreasoningSoap Operación: get_DIAGNOSTICPROCESS_REASONING	9
30	WSDL: BookReaderreview.wsdl Servicio: BookReaderreviewSoap Operación: get_READER_REVIEW	19
31	WSDL: geocodeUSAddress.wsdl Servicio: AddressLocationSoap Operación: get_LATITUDE_LONGITUDE	20
32	WSDL: FranceMap.wsdl Servicio: MapSoap Operación: get_MAP	6
33	WSDL: _hotel_Worldwideservice.wsdl Servicio: HotelSoap Operación: get_HOTEL	4
34	WSDL: findNearbyWikipediaArticles.wsdl Servicio: NearbyWikipediaArticlesSoap Operación: get_LATITUDE_LONGITUDE_DISTANCE_WIKIPEDIA-ARTICLE	29
35	WSDL: _camerataxedpricedutytax_service.wsdl Servicio: _CamerataxedpricedutytaxSoap Operación: get_CAMERA_TAXEDPRICE_DUTYTAX	6
36	WSDL: untangibleobjects_cola_service.wsdl Servicio: UntangibleobjectsColaSoap Operación: get_COLA	9
37	WSDL: BookPricesizebook-type.wsdl Servicio: BookPricesizebook-typeSoap Operación: get_PRICE_SIZE_BOOK-TYPE	24
38	WSDL: Geographical-regionDrought.wsdl Servicio: Geographical-regionDroughtSoap Operación: get_DROUGHT	9

39	WSDL: bookperson_service.wsdl Servicio: BookpersonSoap Operación: get	16
40	WSDL: unlock_door.wsdl Servicio: UnlockdoorSoap Operación: get	6
41	WSDL: findPlaceNamePostalCode.wsdl Servicio: PostalCodesPlacesSoap Operación: get_COUNTRYCODE_LONGITUDE_POSTALCODE_PLACENAME_LATITUDE	22
42	WSDL: _author_DMservice.wsdl Servicio: AuthorSoap Operación: get_AUTHOR	4
43	WSDL: university_academic-support-staff_service.wsdl Servicio: UniversityAcademic-support-staffSoap Operación: get_ACADEMIC-SUPPORT-STAFF	15
44	WSDL: BookPrice.wsdl Servicio: BookPriceSoap Operación: get_PRICE	18
45	WSDL: _digitalstandardpriceprice_MediaMarktservice.wsdl Servicio: DigitalstandardpricepriceSoap Operación: get_DIGITAL_STANDARD_PRICE_PRICE1	7
46	WSDL: BookPerson.wsdl Servicio: BookPersonSoap Operación: get_PERSON	16
47	WSDL: Tizonbook_recommendedpriceindollar_service.wsdl Servicio: BookRecommendedpriceindollarSoap Operación: get_RECOMMENDEDPRICEINDOLLAR	16
48	WSDL: flip_down_slider.wsdl Servicio: FlipdownsliderSoap Operación: getSliderCupUltraSoundSensorContactSensorCarriageCarriageSpatialZone	24
49	WSDL: AcademicAddress.wsdl Servicio: AcademicAddressSoap Operación: get_ADDRESS	26

50	WSDL: TitleCdpricesoftware.wsdl Servicio: TitleCdpricesoftwareSoap Operación: get_CD_PRICE_SOFTWARE	18

Cuadro A.1: Conjunto de pruebas.

Bibliografía

[1] Jorge Cardoso, Alistair P. Barros, Norman May, and Uwe Kylau. Towards a unified service description language for the internet of services: Requirements and first developments. In *2010 IEEE International Conference on Services Computing, SCC 2010, Miami, Florida, USA, July 5-10*, pages 602–609. IEEE Computer Society, 2010.

[2] Carlos Pedrinaci and John Domingue. Toward the next wave of services: Linked services for the web of data. *J.UCS Journal of Universal Computer Science*, 16(13):1694–1719, 2010.

[3] Abdali Mohammadi Fardin, Nemat Bakhsh Naser, and Nematbakhsh Mohammad Ali. Empower service directories with knowledge. *Knowledge-Based Systems*, 30:172–184, 2012.

[4] Z. Maamar, P. Santos, L. Wives, Y. Badr, N. Faci, and J.P.M. de Oliveira. Using social networks for web services discovery. *Internet Computing, IEEE*, 15(4):48–54, 2011.

[5] Lin Lin and I. Budak Arpinar. Discovery of semantic relations between web services. In *Web Services, 2006. International Conference on Web Services(ICWS'06), Chicago, USA*, pages 357–364, September 18-22, 2006.

[6] Eleni Stroulia and Yiqiao Wang. Structural and semantic matching for assessing web-service similarity. *International Journal of Cooperative Information Systems*, 14(4):407–438, 2005.

[7] Mehmet Senvar and Ayse Basar Bener. Matchmaking of semantic web services using semantic-distance information. In Tatyana M. Yakhno and Erich J. Neuhold, editors, *Advances in Information Systems, 4th International Conference, ADVIS 2006, Izmir, Turkey, October 18-20, 2006, Proceedings*, volume 4243 of *Lecture Notes in Computer Science*, pages 177–186. Springer, 2006.

[8] Akin Günay and Pinar Yolum. Semantic matchmaking of web services using model checking. In Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons, editors, *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 1*, pages 273–280. IFAAMAS, 2008.

[9] Leonard Richardson and Sam Ruby. *RESTful Web Services*. O'Reilly, Beijing, 2007.

[10] A. Sahai and S. Graupner. *Web Services in the Enterprise: Concepts, Standards, Solutions, and Management*. Network and Systems Management. Springer, 2007.

[11] L. Dikmans and R. Van Luttikhuizen. *SOA Made Simple*. Professional expertise distilled. Packt Publishing, Limited, 2012.

[12] Vikas Agarwal, Girish Chafle, Sumit Mittal, and Biplav Srivastava. Understanding approaches for web service composition and execution. In R. K. Shyamasundar, editor, *Proceedings of the 1st Bangalore Annual Compute Conference, Compute 2008, Bangalore, India, January 18-20, 2008*, page 8. ACM, 2008.

[13] Biplav Srivastava and Jana Koehler. Web service composition - current solutions and open problems. In *In: ICAPS 2003 Workshop on Planning for Web Services*, pages 28–35, 2003.

[14] Zhengang Cheng, Munindar P. Singh, and Mladen A. Vouk Y. Composition constraints for semantic web services. In *WWW2002 Workshop on Real World RDF and Semantic Web Applications*, page 10, 2002.

[15] Akin Günay and Pinar Yolum. Structural and semantic similarity metrics for web service matchmaking. In Giuseppe Psaila and Roland Wagner, editors, *E-Commerce and Web Technologies, 8th International Conference, EC-Web 2007, Regensburg, Germany, September 3-7, 2007, Proceedings*, volume 4655 of *Lecture Notes in Computer Science*, pages 129–138. Springer, 2007.

[16] J. Hsu. *Multiple Comparisons: Theory and Methods*. Guilford School Practitioner. Taylor & Francis, 1996.

[17] I. Kaliszewski. *Soft Computing for Complex Multiple Criteria Decision Making*. International Series in Operations Research & Management Science. Springer, 2006.

[18] E.R. Harold. *XML 1.1 bible*. Bible Series. Wiley Pub., 2004.

[19] A. Skonnard and M. Gudgin. *Essential Xml Quick Reference: A Programmer's Reference to Xml, Xpath, Xslt, Xml Schema, Soap, and More*. The DevelopMentor series. ADDISON WESLEY Publishing Company Incorporated, 2002.

[20] S. Powers. *Practical RDF*. O'Reilly Media, 2003.

[21] S.M. Solís. *La Web Semántica*. Lulu Enterprises Incorporated, 2007.

[22] Douglas K. Barry and Patrick J. Gannon. *Web Services and Service-Oriented Architecture: The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[23] E. Newcomer. *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Independent technology guides. Addison-Wesley, 2002.

[24] K. Topley. *Java Web Services in a Nutshell*. In a Nutshell (o'Reilly) Series. OReilly, 2003.

[25] J. Bean. *SOA and Web Services Interface Design: Principles, Techniques, and Standards*. The MK/OMG Press. Elsevier Science, 2009.

[26] D. Fensel, F.M. Facca, E. Simperl, and I. Toma. *Semantic Web Services*. Springer, 2011.

[27] Beaver Mendenhall, W. *Introducción A la Probabilidad Y Estadística*. Cengage Learning Latin America, 2010.

[28] S.F. Fernández, J.M.C. Sánchez, A. Córdoba, J.M. Cordero, and A.C. Largo. *Estadística descriptiva*. Libros profesionales. Esic Editorial, 2002.

[29] S. Holzner. *Inside XML*. Inside (New Riders). New Riders, 2001.

[30] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, December 1989.

[31] Philip N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, ESA '98, pages 91–102, London, UK, UK, 1998. Springer-Verlag.

[32] Pedro Morales Vallejo. *Estadística aplicada a las Ciencias Sociales*. Universidad Pontificia Comillas, 2008.

[33] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Estructuras de datos y algoritmos*. Addison-Wesley Iberoamericana, 1988.

[34] I. Hlavats. *Jsf 1.2 Components*. From technologies to solutions. Packt Publishing, Limited, 2009.

[35] Sergio Jimenez, Claudia Becerra, Alexander Gelbukh, and Fabio Gonzalez. Generalized mongue-elkan method for approximate text string comparison. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 5449 of *Lecture Notes in Computer Science*, pages 559–570. Springer Berlin Heidelberg, 2009.