Middleware para el Descubrimiento, Composición y Ejecución Automática de Servicios Web Semánticos Especificados con el Framework FODAS-WS

Jose Aguilar, Omar Portilla
Centro de Estudios en Microelectrónica y Sistemas Distribuidos (CEMISID)
Universidad de Los Andes
Mérida, Venezuela
aguilar@ula.ve, oportillajaimes@yahoo.es

***Resumen*—Este trabajo presenta la arquitectura general de un middleware para el descubrimiento, la composición y la ejecución automática de servicios web semánticos, especificados en el marco de trabajo FODAS-WS, llamado MDCE-FODAS. MDCE-FODAS se implementa como un servicio web que permite realizar consultas desde cualquier aplicación cliente, y su ejecución se realiza de forma automática y transparente por parte de MDCE-FODAS. MDCE-FODAS permite automatizar la gestión de los servicios web semánticos en una aplicación SOA. Este artículo describe la arquitectura y presenta ejemplos de aplicaciones.**

Palabras Clave—FODAS-WS, Servicios Web Semántico, Middleware, Aplicaciones Orientadas a Servicios

I. Introducción

Este trabajo presenta un Middleware para el descubrimiento, composición y ejecución automática de servicios web semánticos (SWS), llamado MDCE-FODAS. MDCE-FODAS se inserta en la arquitectura del Framework FODAS-WS propuesto en [1], en el componente WS-GENERADOR-CLIENTE. MDCE-FODAS se implementa como un servicio web que encapsula un conjunto de servicios proporcionados a clientes que soliciten la ejecución automática de necesidades, usando el protocolo HTTP. La solicitud de la ejecución automática de las necesidades no requiere que dicho cliente conozca la arquitectura y funcionamiento del Framework FODAS-WS.

MDCE-FODAS utiliza como insumo un repositorio de especificaciones semánticas de SWS realizadas con FODAS-WS [1]. Cada uno de los SWS especificados en tal repositorio, se ajustan a la arquitectura planteada en el modelo de referencia SOA. MDCE-FODAS realiza el emparejamiento entre las necesidades del cliente y los SWS descritos con FODAS-WS. Para llevar a cabo tal tarea, MDCE-FODAS encapsula como servicio tres procesos que se realizan de forma automática, que son: el Descubrimiento, la Composición y la Ejecución.

Las principales características de MDCE-FODAS son: a) Oculta la complejidad de los procesos de descubrimiento composición y ejecución automática de SWS especificados con FODAS-WS; b) Proporciona una interfaz que permite que cualquier aplicación web cliente ejecute automáticamente las necesidades que requiera; c) Proporciona un conjunto de servicios comunes que reduce la duplicación de esfuerzos para realizar los procesos de descubrimiento composición y ejecución automática de SWS especificados con FODAS-WS.

Este artículo se divide en seis secciones: la Sección II presenta trabajos relacionados; en la Sección III es presentado el marco de referencia de este trabajo, como es el Framework FODAS-WS; la Sección IV se encarga de presentar a MDCE-FODAS, su arquitectura y la descripción de sus componentes; la Sección V presenta dos casos de estudio, y finalmente, la Sección VI describe las conclusiones.

II. Trabajos Relacionados

En [2] se presenta una aproximación semántica basada en ontologías para el descubrimiento automatizado de servicios web. Los servicios web se crean y almacenan en el registro de servicios. Los autores proponen la creación de una ontología de dominio que apoye el descubrimiento eficiente de los servicios, basados en la solicitud del usuario. Se utiliza un algoritmo de emparejamiento para la búsqueda de coincidencias, entre los servicios cuya descripción esta almacenada en la ontología y el servicio solicitado.

En [3] diseñan e implementan un kit generador de servicios (SGT) que permite a investigadores crear grandes cantidades de servicios web. El SGT genera automáticamente servicios web a partir de un modelo gráfico abstracto de servicios, y luego convierte ese modelo en código.

En [4] presentan Orinoco, una infraestructura para ambientes Grid que integra las facilidades de publicación, descubrimiento, composición y ejecución de servicios web. Ese trabajo permite transformar aplicaciones JAVA en servicios web, generando las descripciones semánticas correspondientes. También asiste a los usuarios, resolviendo requerimientos que impliquen la coordinación de servicios. Ante un requerimiento, Orinoco genera un plan de ejecución que dará respuesta al requerimiento. En el trabajo demuestran las funcionalidades de Orinoco con un caso de estudio, y se presentan resultados de rendimiento sobre gLite, un middleware de Grid.

En [5] proponen la composición automática de servicios web basados en efectos de ejecución inciertos. Para ello, acuden a la técnica de planificación de Inteligencia Artificial Graphplan. En el trabajo se presentan casos de prueba, y se describen trabajos relacionados. En [6] explican un algoritmo de búsqueda basado en heurísticas que realiza la composición automática de servicios web. En el trabajo se utiliza el Algoritmo A* para resolver el problema de emparejamiento semántico durante la composición de servicios web. Para ello, dada una solicitud, se genera un grafo de dependencia de servicio en forma dinámica.

En [7] proponen un framework para el descubrimiento automático de servicios web basado en semántica y técnicas de procesamiento de lenguaje natural. En ese trabajo, el descubrimiento es realizado empleando técnicas del procesamiento de lenguaje natural para emparejar las necesidades expresadas en lenguaje natural.

Por otro lado, en [12] diseñan un Medio de Gestión de Servicios para Sistemas Multiagentes. Por otra parte, en [13] proponen una arquitectura de middleware para el diagnóstico de composiciones de servicio completamente distribuidas, llamada ARMISCOM, que no está coordinada por ningún diagnosticador global. El diagnóstico de fallas se realiza a través de la interacción del diagnosticador presente en cada composición del servicio, y las estrategias de reparación se desarrollan a través del consenso de cada reparador distribuido en cada composición del servicio.

Finalmente, [14] presenta un middleware que permite que aplicaciones basadas en agentes y en servicios web se comuniquen de manera bidireccional y transparente, y, además, utiliza el paradigma de la computación de niebla de forma autónoma, según el contexto y la carga del sistema.

El trabajo presentado en este artículo se diferencia de los anteriores en que es el único que presenta un middleware que permite realizar el proceso completo de descubrimiento, composición y ejecución automática de servicios web, en forma automática y transparente para el usuario que lo solicita. En ninguno de los trabajos analizados se propone un framework para realizar una descripción semántica completa de los servicios. Este trabajo presenta mecanismos que permiten procesos de generación automática de aplicaciones a partir de sus especificaciones semánticas, reutilizando servicios web existentes.

III. FODAS-WS

En [1] se presenta FODAS-WS, una herramienta para la realización del diseño de servicios web basada en ODA (Ontology Driven Architecture) [8, 9]. El Framework proporciona un nivel de definición semántica, que puede soportar procesos de composición automática, con la generación automática del código de la aplicación cliente.

FODAS-WS busca contribuir en el proceso de descubrimiento, emparejamiento y composición automática de servicios, mediante modelos ontológicos comprensibles a la máquina, almacenados en tres capas de distinto nivel de abstracción basados en el paradigma ODA: CAPACIM, CAPAPIM y CAPAPSM [1]. Cada uno de los modelos ontológicos de las distintas capas de FODAS-WS se almacenan en lenguaje OWL.

La Figura 1 muestra la arquitectura de FODAS-WS. En la capa CIM el framework incluye una subdivisión de tres capas, que se denominan CAPA DE DEFINICION DE DOMINIO (especifica el dominio de la aplicación), CAPA CIM REFERENCIAL (contempla los referentes arquitectónicos) y la CAPA CIM OPERACIONAL (describe las funcionalidades de la aplicación). Esta capa sirve de insumo para que el middleware realice el proceso de descubrimiento, composición y ejecución automático de servicios web de acuerdo a las necesidades del cliente.

En la CAPA DE DEFINICION DE DOMINIO se especifica todo el conocimiento para describir un dominio. La descripción de cada dominio es realizada en torno al concepto de ASOCIACION. Una ASOCIACION es una tripleta que relaciona dos ELEMENTOS mediante una ACCION. Cada uno de los ELEMENTOS especificados en el dominio son clases de individuos que intervienen, y una ACCION es un tipo de intervención, trabajo, modificación, influencia, que un elemento realiza sobre otro. Por ejemplo, como parte de la especificación de un dominio de librería puede existir una ASOCIACION que determine que a los lectores les gusta cierto género literario. Para ello, en el dominio se debe especificar una ASOCICACION llamada aso:lector_genero en la que un ELEMENTO lector (antecedente), ejecuta la ACCION gustar sobre el ELEMENTO género(consecuente).

Es importante resaltar que la ASOCIACIÓN es fundamental, porque, es en torno a ellas que se da el proceso de descripción de las CAPACIDADES y NECESIDADES, de igual forma que los efectos, las precondiciones y los estados. Es decir, tanto una CAPACIDAD como una NECESIDAD son una ASOCIACION previamente especificada en el dominio.

La capa CIMREFERENCIAL articula el conocimiento especificado en las distintas capas de FODAS-WS. Esta capa cuenta con un modelo ontológico, en el que, además de realizar una descripción arquitectónica, se describen las capacidades del servicio. Específica, además, las precondiciones y efectos asociadas con cada capacidad. Especificar una CAPACIDAD implica relacionarla con una ASOCIACION que se encuentre especificada en la capa de definición del dominio

La capa PIM representa el segundo nivel de abstracción en FODAS-WS, al expresar en forma abstracta la estructura a usar en la implementación del servicio. La especificación de esta capa es realizada automáticamente mediante un proceso de transformación de CIMPIM descrito en [1]. El conocimiento almacenado en la CAPAPSM es generado automáticamente por la transformación PIM-PSM explicada en [1]. Esta capa ofrece detalles para la implementación del servicio.

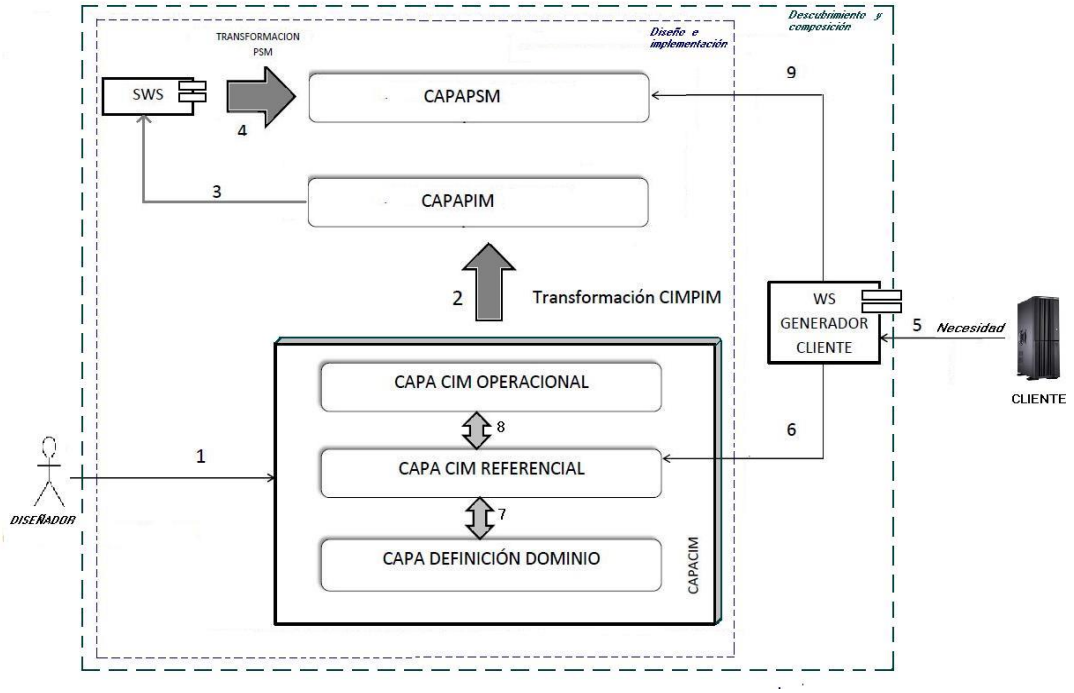

Figura 1. Arquitectura de FODAS-WS Presentada en [1]

IV. Arquitectura del MDCE-FODAS

A. Generalidades

MDCE-FODAS cuenta con procesos que permitan suplir las necesidades del cliente con las capacidades ofrecidas por los servicios web descritos con FODAS-WS. Para llevar a cabo tal tarea, MDCE-FODAS encapsula como servicio tres procesos:

- Descubrimiento Automático: MDCE-FODAS recibe como parámetro de entrada la NECESIDAD solicitada por la aplicación cliente. Seguidamente, lleva a cabo un proceso de emparejamiento, considerando todas las CAPACIDADES ofrecidas por los distintos servicios web con que se cuenta en el repositorio. El proceso de emparejamiento construye un conjunto de emparejamiento, priorizado según una función de emparejamiento calculada por MDCE-FODAS. Este proceso termina al determinar cuál es la CAPACIDAD que mejor empareja con la NECESIDAD solicitada.
- Composición Automática: MDCE-FODAS determina el flujo de operaciones que se debe invocar para lograr la capacidad que mejor emparejó en el proceso de descubrimiento automático. Para ello, hace uso de las especificaciones del servicio realizadas en las distintas capas de FODAS-WS, especialmente, lo especificado en la capa CIM OPERACIONAL.
- Ejecución Automática: Consiste en generar aplicaciones cliente para los servicios web involucrados en la ejecución del flujo de operaciones determinado durante el proceso de composición automática. Para generar tales aplicaciones clientes, es necesario que MDCE-FODAS acuda a lo especificado en la capa PSM de cada uno de los servicios involucrados. Una vez generado y compilado el código de las aplicaciones cliente de los servicios involucrados, se procede a invocar las operaciones asociadas a cada servicio, y a transferir las respuestas obtenidas a la aplicación que solicitó la necesidad al MDCE-FODAS.

B. Arquitectura de Base

La Figura 2 ilustra cómo MDCE-FODAS sirve de intermediario entre los servicios web (SWS) descritos en FODAS-WS, y los clientes que solicitan necesidades.

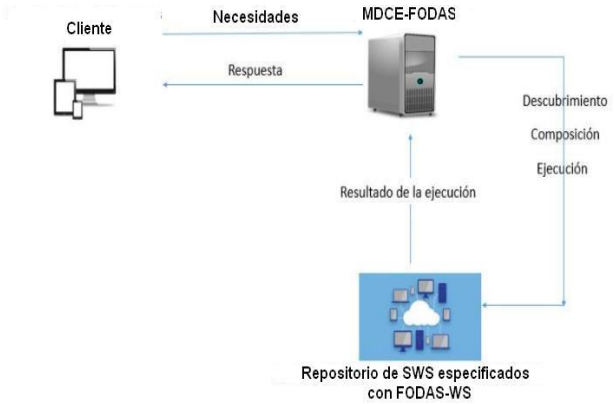

Figura 2. Descripción General del Proceso de Intermediación Realizado por MDCE-FODAS

MDCE-FODAS cuenta con una arquitectura multicapa compuesta por tres capas (ver Figura 3).

1) Capa de Acceso a la Información Semántica

Esta capa contiene la implementación necesaria para acceder a los diversos modelos ontológicos de FODAS-WS. Interactúa directamente con el repositorio de especificaciones semánticas

de SWS. Esta capa garantiza independencia del MDCE-FODAS, ante posibles modificaciones en los modelos ontológicos del FODAS-WS

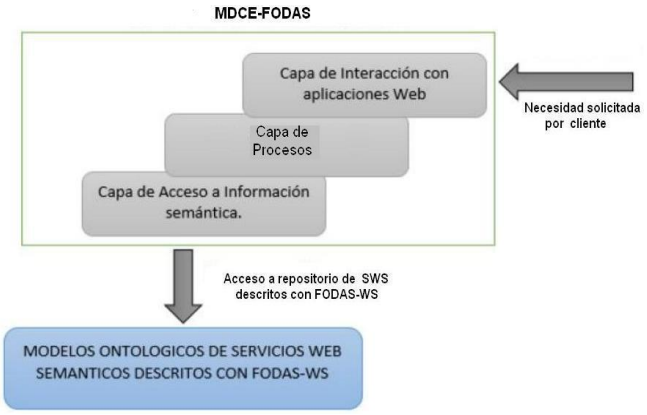

Figura 3. Arquitectura de MDCE-FODAS

2) Capa de Procesos

Esta capa implementa los tres procesos fundamentales llevados a cabo por MDCE-FODAS, los cuales son: descubrimiento, composición y ejecución automática. La capa cuenta con los siguientes cuatro componentes fundamentales, cuya interacción es mostrada en la Figura 5: núcleo, compositor, administrador dinámico de servicios web, y ejecutor.

Figura 4. Capa de Interacción de Aplicaciones Web

- *Núcleo:* Realiza el proceso de descubrimiento automático de las capacidades asociadas con los SWS. Además, se encarga de gestionar las comunicaciones con los demás componentes de la capa, así como con las otras capas que conforman la arquitectura de MDCE-FODAS.
- *Compositor:* Realiza el proceso de composición automática. Para ello, construye una lista de operaciones necesarias, indicando el orden en que deben ser invocadas, los parámetros de cada una, su tipo de retorno. Además, determina el SWS a que pertenece cada operación.
- *Administrador Dinámico de Servicios Web:* Crea en forma automática el código para acceder a las capacidades ofrecidas por los SWS. Para ello, el compositor le proporciona una lista de los SWS que se requieren. Para cada SWS requerido, este componente genera código automático del cliente usando la herramienta wsimport, que toma como insumo de entrada el archivo WSDL de cada SWS.
- *Ejecutor:* Invoca las operaciones en el orden determinado por el compositor, usando el código generado por el administrador dinámico de servicios web. Una vez obtenida la respuesta de los SWS invocados, este componente se encarga de parsear la respuesta a HTML y entregarla al núcleo, para que este se la envié a la aplicación cliente.

3) Capa de Interacción con Aplicaciones Web

Esta capa se encarga de recibir la NECESIDAD desde la aplicación cliente, pero, además, una vez terminado el proceso de ejecución automática, le envía los resultados al cliente solicitante. La Figura 4 muestra las interacciones de dicha capa. Es importante aclarar que la NECESIDAD N recibida del cliente es una ASOCIACION como las descritas en la Sección III.

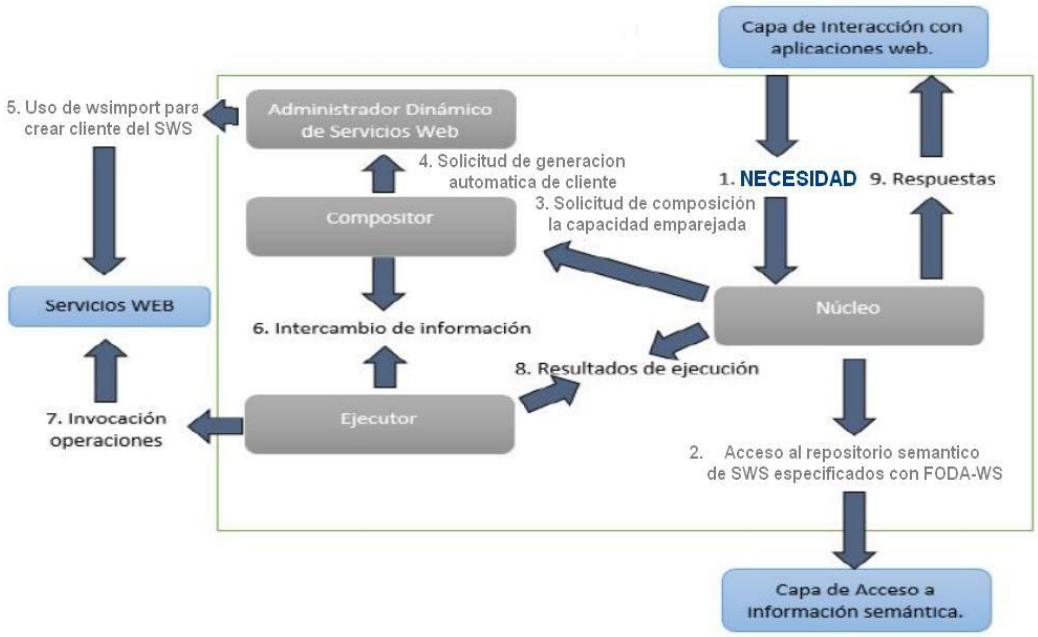

Figura 5. Procesos Desarrollados por la Capa de Procesos

C. Procesos Realizados por MDCE-FODAS

1) Proceso de Descubrimiento Automático de SWS

La Figura 6 muestra el proceso de descubrimiento automático llevado a cabo por el núcleo de MDCE-FODAS. Dicho proceso se compone de tres subprocesos fundamentales:

- Construcción del conjunto de recomendación
- Calculo del grado de emparejamiento de cada objeto CAP_i
- Rankeo del conjunto de recomendación

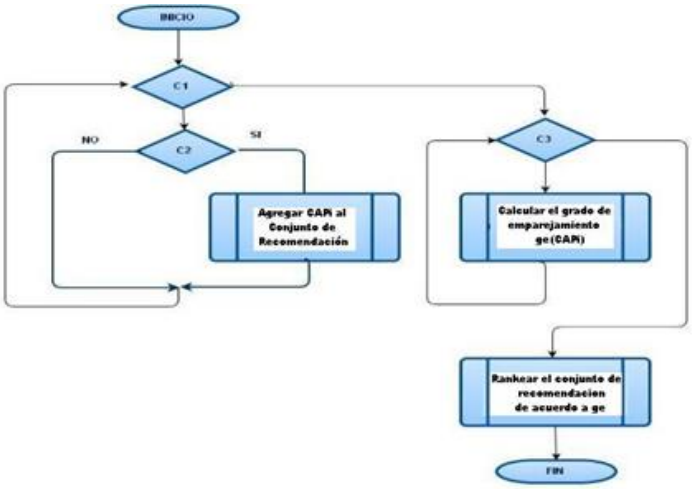

Figura 6. Proceso de Descubrimiento Automático

a) Subproceso de Construcción del Conjunto de Recomendación

Consiste en construir el conjunto de capacidades que emparejan con la necesidad solicitada, al que se le llamará CONJUNTO DE RECOMENDACION. El conjunto de recomendación se compone de elementos llamados CAMINO NECESIDAD-CAPACIDAD. Cada CAMINO NECESIDAD-CAPACIDAD se conforma de una o más ASOCIACIONES, que buscan enlazarse con una CAPACIDAD CAP_i que de alguna forma pueda satisfacer la NECECIDAD solicitada por el cliente. Cada uno de estos caminos tiene un número de los componentes del camino Nc. Nc indica el número de ASOCIACIONES que integran el camino. Un camino con Nc=1 indica que existe una CAPACIDAD que empareja directamente con la NECESIDAD solicitada. Decir que la CAPACIDAD empareja directamente con la NECESIDAD implica que el sistema de inferencias del MDCE-FODAS determinó que la ASOCIACION que representa la CAPACIDAD tiene similaridad con la ASOCIACION que representa la NECESIDAD. En caso que Nc tenga un valor mayor que 1, implica que no se encontró una CAPACIDAD que empareja directamente con la NECESIDAD, lo que exige que el sistema de inferencias construya un camino de asociaciones partiendo de una ASOCIACION semilla que empareje con la NECESIDAD (pero que no representa una CAPACIDAD de ningún servicio). La construcción del camino se da por terminada en el momento en que la última ASOCIACION agregada al camino tenga similaridad con una CAPACIDAD existente. De lo anterior es importante resaltar que cada uno de los caminos que pertenecen al CONJUNTO DE RECOMENDACION tienen una y solo una CAPACIDAD CAP_i asociada.

b) Subproceso de Cálculo del Grado de Emparejamiento para cada Capacidad del Conjunto de Recomendación

Consiste en calcular un valor numérico que se denomina *grado de emparejamiento*, cuyo valor oscila entre 0 y 2.5. Cada capacidad CAP_i asociada con cada camino que pertenezca al conjunto de recomendación tendrá su propio valor para el grado de emparejamiento. Dicho valor representa lo adecuada que es la CAPACIDAD CAP_i para responder la NECESIDAD N solicitada por el cliente U_k un valor de 2.5 en el grado de emparejamiento de la CAP_i indica que dicha CAPACIDAD empareja perfectamente con la NECESIDAD solicitada y un valor de 0 en el grado de emparejamiento indica que CAP_i no tiene ningún grado de relación con la NECESIDAD solicitada.

La ecuación (1) muestra la definición formal del grado de emparejamiento (g), la cual es usada por MDCE-FODAS para calcular el grado de emparejamiento que tiene cada capacidad CAP_i que pertenezca al conjunto de recomendación. El grado de emparejamiento es una función cuyo dominio es el producto cartesiano del conjunto de capacidades C y el conjunto de usuarios U. El valor calculado por el grado de emparejamiento es de vital importancia, pues es el criterio mediante el cual MDCE-FODAS determina la capacidad más idónea para suplir la necesidad solicitada por el cliente U_k.

$$g(CAP_i, U_k, N) = Na(CAP_i, N, U_k) + Lc(CAP_i, N) + Ne(CAP_i, U_k) \qquad (1)$$

dónde:

$g(CAP_i, U_k, N)$ es el grado de emparejamiento que la capacidad CAP_i tiene respecto de la necesidad N solicitada por el usuario U_k.

$Na(CAP_i, N, U_k)$ es el nivel de acople de la necesidad solicitada. Esta función retorna un valor comprendido entre 0 y 1, que estipula que tanto la necesidad N solicitada se acopla con la capacidad CAP_i en el dominio al que pertenece dicha capacidad.

$Lc(CAP_i, N)$ retorna un valor comprendido entre 0 y 1. La ecuación (2) muestra la forma de calcular *Lc*, donde Nc es una función que retorna el número de componentes para dicho camino necesidad_capacidad.

$$Lc(CAP_i, N) = 1/Nc\,(CAP_i, N) \quad (2)$$

$Ne(CAP_i, U_k)$Es una función que calcula el nivel de enfoque *Ne*, entregando un valor entre 0 y 0.5. 0.5 indica que las CAP_i está relacionada directamente con el enfoque de mayor interés para el usuario U_k.

c) Subproceso de Rankeo del Conjunto de Recomendación

Constituye la fase final del proceso de descubrimiento automático, y fundamentalmente, consta de un ordenamiento realizado a las capacidades que conforman el conjunto de recomendación, usando como criterio de ordenamiento el grado de emparejamiento calculado en el subproceso anterior.

2) Proceso de Composición Automático de SWS

Este proceso toma como insumo de entrada el CONJUNTO DE RECOMENDACIÓN construido en el proceso de descubrimiento automático. De las distintas CAPACIDADES asociadas con el CONJUNTO DE RECOMENDACIÓN, se elige una que en adelante se llamará C, dicha CAPACIDAD es la CAP_i que tenga mayor grado de emparejamiento. Es importante recordar que en [1] se estipula que C debe tener vinculado al repositorio las especificaciones semánticas de las capas CIMOPERACIONAL, PIM y PSM. A partir de tales especificaciones semánticas, el compositor del MDCE-FODAS genera una lista de operaciones a ejecutar, en la que se considera: el orden de ejecución de las operaciones, los argumentos de entrada de cada operación, los tipos de datos a retornar por cada operación, y el SWS en el que dicha operación está implementada.

Para la conformación de la lista de operaciones a ejecutar, el compositor del MDCE-FODAS tiene en cuenta que actividades están asociadas con C en la especificación de la CAPA CIM OPERACIONAL de C, y el orden de dichas actividades.

Con la lista de operaciones construida (LOC), el compositor solicita al administrador dinámico de servicios web que genere automáticamente el código necesario para invocar las operaciones incluidas en LOC. Para ello, crea una lista de administradores dinámicos de servicios (LADS), que se asocian uno a uno con cada SWS incluidos en LOC. Cada uno de los objetos incluidos en LADS debe contener el código generado para la ejecución del SWS, de acuerdo a las descripciones dadas en el archivo WSDL asociado a él. La generación automática de los códigos necesarios se realiza mediante el uso de la herramienta wsimport.

3) Proceso de Ejecución Automático de SWS

Este proceso requiere como entrada las listas LOC y LADS construidas en el proceso de composición automática de SWS. El ejecutor del MDCE-FODAS realiza la ejecución automática de cada una de las operaciones incluidas en LOC, en el orden en que se encuentren registradas de manera secuencial. Para ello, invoca los métodos incluidos en los códigos generados automáticamente de los servicios que conforman LADS, y hace uso de las especificaciones semánticas de la CAPA PSM de cada uno de estos servicios.

El Ejecutor crea un HashMap (diccionario), llamado pilaValores, en el cual se guardan los valores retornados por cada una de las operaciones, para garantizar que los valores retornados por las operaciones puedan ser usados como parámetros de entrada en otra operación en caso de que lo requiera.

Una vez concluida la ejecución automática de la capacidad, el ejecutor ejecuta un parser HTML para entregar los resultados obtenidos en dicho formato.

V. EXPERIMENTOS CON MDCE-FODAS

Para la realización de las pruebas de funcionamiento se emuló una aplicación cliente, a partir de la cual se solicitan necesidades. Siguiendo lo indicando en [1], cada necesidad se especifica de la forma antecedente-acción-consecuente.

Como se explicó en la Sección III-B, MDCE-FODAS requiere un repositorio de descripciones semánticas en las que se incluyen especificaciones del dominio que describen las distintas capacidades que los servicios ofrecen. Para ilustrar una prueba sencilla se puede observar la Figura 7, que especifica las operaciones necesarias para ejecutar la capacidad DA_promediar, perteneciente al dominio operaciones Aritméticas. Como lo muestra la Figura 6, la ejecución de esta capacidad requiere operaciones como realizar_suma, contar_números y dividir_numeros.

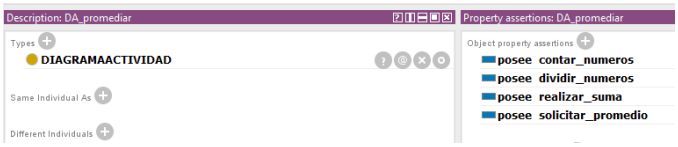

Figura 7. Descripcion Semántica de la Capacidad DA_promediar

Cuando en la aplicación cliente de prueba se solicita la necesidad, cuya acción es promediar y cuyo consecuente es la lista de números 20,50,10,40, se obtiene el conjunto de emparejamiento mostrado en la Figura 8.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-numero3 ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 8. Conjunto de Emparejamiento Generado en la Necesidad ‘promediar 20,50,10,40’

Después, MDCE-FODAS construye la lista LOC que contiene la siguiente lista de operaciones: sumar, contar y dividir en su respectivo orden. La Figura 9 muestra la forma en que el ejecutor procede a invocar la operación sumar, recibiendo como respuesta el valor de 120. Es importante resaltar que el método sumar no se encuentra implementado dentro del servicio web WS_promedio que calcula el promedio, sino dentro de otro servicio web llamado WS_Suma. Esto evidencia la capacidad de composición y ejecución automática de MDCE-FODAS, pues,

para ello, debió no solo componer los métodos, sino generar código automático que permita invocar el método sumar en dicho servicio.

```
Información:   Agregando operacion a ejecutar sumar
Información:   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Información:   Entra a ejecutar la operacion _ sumar
Información:   creando Object[] de tipo int[]
Información:   Agregando parametro 20
Información:   Agregando parametro 50
Información:   Agregando parametro 10
Información:   Agregando parametro 40
Información:   llamando operacion en el servicio operaciones
Información:   Buscando metodo sumar
Información:   Encontró el metodo _
Información:   resultado respuesta numero : 120
Información:   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Información:   ANALIZANDO : numero -> 120
```

Figura 9. Invocacion de la Operacion Suma por Parte del Ejecutor

Un proceso similar realiza el ejecutor para la operación contar que se encuentra en el web service WS_contar. Una vez obtenido el resultado de las operaciones sumar y contar, el ejecutor utiliza los valores obtenidos para calcular el promedio de los números mediante la invocación de la operación dividir, que se encuentra disponible dentro de una de las capacidades del servicio WS_dividir.

Terminada la ejecución de la operación calcular_promedio, se llama al método parsearAHTML que analiza cada de los resultados de la pila de valores y los pone en formato HTML. El resultado mostrado en la Figura 10 es el arrojado cuando el cliente solicita 'promediar 20,50,10,40'.

Resultados
120
4
30.0

Figura 10. Resultados Mostrados para la Necesidad 'promediar 20,50,10,40'

Hasta acá, se ha descrito un sencillo caso, en el que para calcular el promedio de los números 20,50,10 y 40, MDCE-FODAS debe componer y ejecutar en forma automática tres servicios web distintos. Ahora se presentará un nuevo ejemplo que clarifique lo asociado a la manera de especificar semánticamente los servicios, y a como se desarrolla el proceso de descubrimiento, composición y ejecución automática de dicho servicio.

La Figura 11 ilustra el modelado del dominio biblioteca. Dentro de la definición semántica de acciones y elementos en dicho dominio, se han especificado las relaciones conceptuales de los mismos. Por ejemplo, en la Figura 12 se observa los conceptos semejantes al concepto de autor.

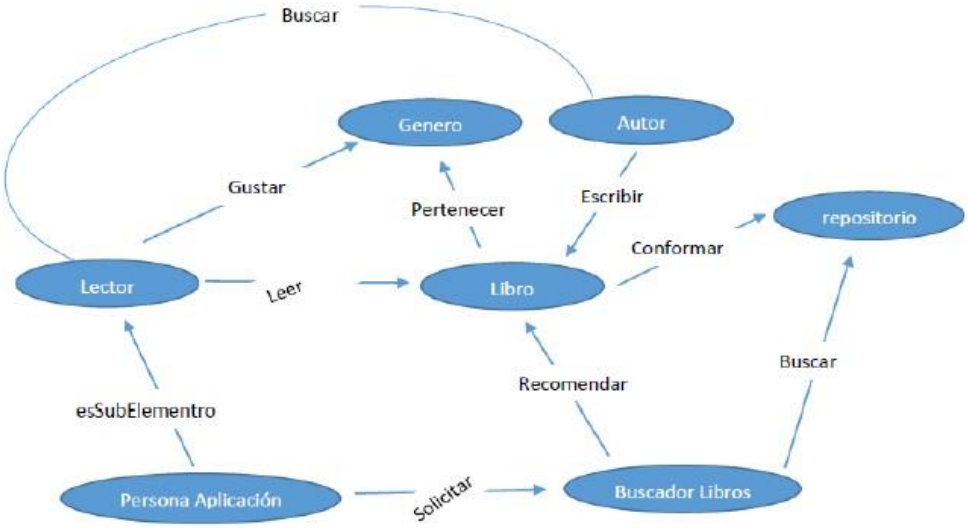

Figura 11. Representacion del Dominio de Librerías

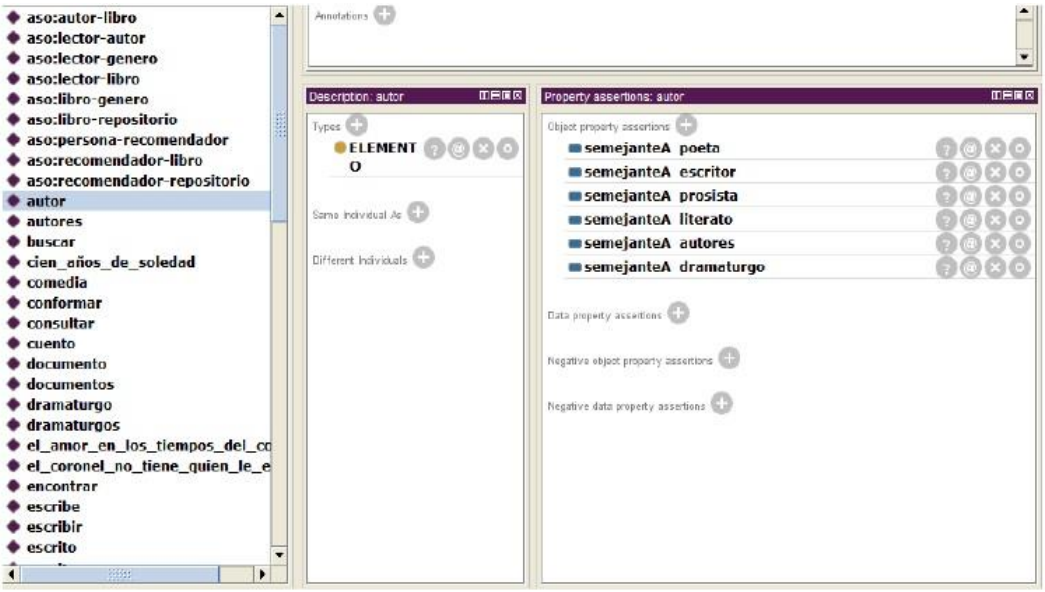

Figura 12. Definición Semántica del Elemento Autor Dentro de la Definición Semántica de Dominio de Librerías

El flujo de proceso modelado para ese domino se muestra en la Figura 13. En el flujo de proceso se introducen tres asociaciones, con su respectiva antecedentes, que indican que el flujo normal inicia cuando una aplicación cliente solicita una NECESIDAD de búsqueda de libros (asociación aso:persona-recomendador), y termina con la recomendación de libros por parte del buscador de libros (asociación aso:recomendador-libro).

Figura 13. Flujo General de Proceso del Dominio Librerías

En adelante se analiza la capacidad de recomendar libros. La capacidad de recomendar libros está representada por la asociación “Buscador Libros - Recomendar – Libro”, contenida en el modelo ontológico de definición del dominio como aso:recomendador-libro. Como se observa en la Figura 14, la capacidad se encuentra registrada en la CAPA CIM REFERENCIAL.

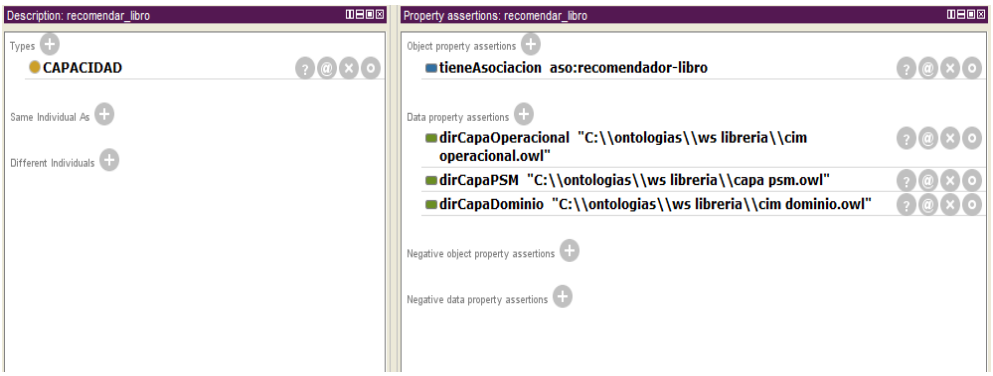

Figura 14. Descripción Semántica de la Capacidad Recomendar Libro

El diagrama de actividad plasmado en la descripción semántica de la capacidad del buscador de libros, modelada en la capa CIM OPERACIONAL, se observa en la Figura 15.

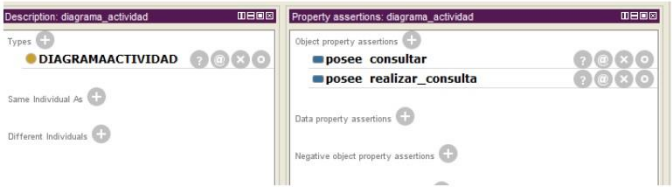

Figura 15. Actividades Dentro del Diagrama de Actividad de la Capacidad de Recomendar Libros

La actividad 'realizar_consulta' es realizada por el actor 'usuario', y no ejecuta alguna operación dentro de algún servicio web. En cambio, la actividad 'consultar' es ejecutada por el actor 'servicio', lo que le indicará al MDCE-FODAS que la operación ejecutada en esta actividad tendrá que ser ejecutada por él.

Dentro de la actividad 'consultar' se definió además su actor, con la operación 'consultarLibros', como se aprecia en la Figura 16.

Figura 16. Descripción Semántica de la Actividad Consultar

Dentro del modelo ontológico correspondiente a la capa PSM, se define el servicio llamado librería donde se le establecen los data properties dirección del archivo WSDL y paquete que corresponde al namespace o paquete donde fue implementado este servicio, tal como se muestra en la Figura 17.

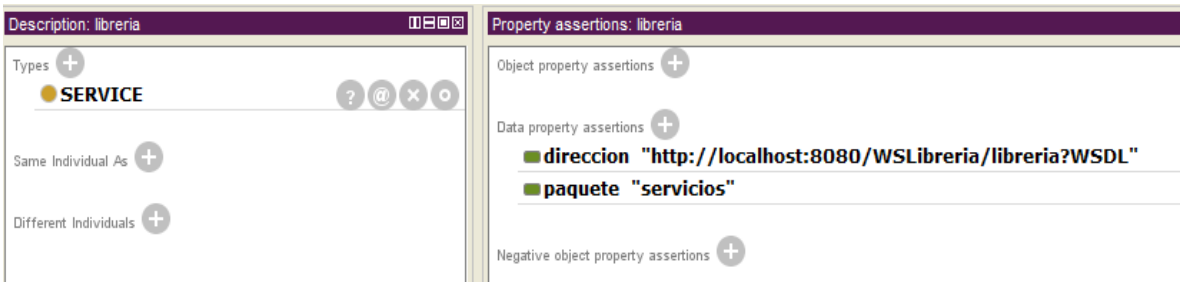

Figura 17. Definición Semántica del Servicio Librería en la Capa PSM

Como se estableció que la ACCION 'leer' empareja directamente con la ACCION 'recomendar', y la ACCION 'recomendar' esta explícitamente definida en la ASOCIACION que expresa la capacidad de recomendar libros, el grado de emparejamiento se espera que sea alto, además de que en la consulta se expresa que lo que se desea leer es libro. El consecuente 'libro' está también directamente asociado a la asociación capacidad. La respuesta al realizar la consulta 'leer libro', es mostrada en la Figura 18.

Resultados

Eric Nylund	Ciencia ficcion	1	Halo: The Fall of Reach	En La caída del Reach se muestra el origen de los SPARTAN-II, escogidos por un indicador genético en su ADN que los hacía aptos para el proyecto Spartan unos 150 niños de los cuales la doctora Halsey acompañada por el futuro Almirante Jacob Keyes, tuvo que seleccionar 75 por falta de presupuesto. Con un entrenamiento especial durante su niñez y un aumento con químicos y partes robóticas se creó una raza de supersoldados o Spartan, cuya finalidad era suprimir levantamientos rebeldes en una futura guerra civil; debido a la guerra contra el Covenant que comenzó en el año 2525 los Spartans fueron enviados a combatir mostrando superioridad a los soldados normales e incluso contra los élites del Covenant (Sangheili). libro. novela.
Christie Golden	Ciencia ficcion	2	Arthas: El Surgir del Rey Exánime	cuenta el ascenso de Arthas Menethil hacia el Trono de Hielo para convertirse en el Rey Exánime, pasando desde su niñez hasta su regreso en Wrath of the Lich King. libro. novela.
Gabriel José de la Concordia García Márquez	Novela	3	Cien años de soledad	El libro narra la historia de la familia Buendía a lo largo de siete generaciones en el pueblo ficticio de Macondo
Rafael Pombo	Poesia	4	Simon el bobito	poesia de simon el bobito. libro.

Figura 18. Respuesta del MDCE-FODAS a la Consulta "leer libro"

MDCE-FODAS envía como respuesta una lista de libros, que es el producto de hacer un llamado a la operación 'consultarLibros' del servicio web librería.

En adelante se describen las acciones realizadas por MDCE-FODAS tras la petición mostrando las salidas internas del sistema. La Figura 19 ilustra los resultados del conjunto de emparejamiento obtenido para la necesidad solicitada. En este

caso, solo la CAPACIDAD aso:recomendador-libro conforma el conjunto de emparejamiento. El grado de emparejamiento de dicha capacidad tuvo un valor de 1.8.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 19. Conjunto de Emparejamiento Generado con la Consulta 'leer libro'

La Figura 20 muestra parte de los procesos de composición automática realizados por MDCE-FODAS. En la Figura 18 se muestra como se extraen las operaciones, sus parámetros y demás aspectos necesarios para construir LOC.

```
Operacion : consultarLibros
parametro caracteristica -> String
parametro libros -> List
Tipo retorno : libros
Primer parametro : caracteristica
Se limpio la lista, nueva 1
Objeto de http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#recomendador
Tiene ATRIBUTO http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#resultados
SE ENCONTRÓ EL DIAGRAMA DE ACTIVIDAD DE LA OPERACIÓN
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#consultar>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#servicio>
Operacion : consultarLibros
parametro caracteristica -> String
parametro libros -> List
Tipo retorno : libros
Primer parametro : caracteristica
Se limpio la lista, nueva 1
Actividad <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#realizar_consulta>
Actor : <http://www.semanticweb.org/admin/ontologies/2014/9/OntoCasoUso#usuario>
Antecede A_: consultar
```

Figura 20. Ejecución de las Operaciones Necesarias para la Capacidad Recomendar Libros

En este caso de prueba, el nivel de enfoque dentro de los elementos de emparejamiento tiene el valor de cero (como se aprecia en la Figura 19), esto debido a que la consulta la realizó un usuario sin tener una sesión activa. Si el usuario inicia una sesión en MDCE-FODAS podrá analizar si el dominio donde está buscando tiene relación con el enfoque del usuario. La Figura 21 muestra el conjunto de emparejamiento para la misma necesidad solicitada, pero realizada por un usuario registrado con un enfoque académico.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 2.3
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:recomendador-libro ->
Información:   1/LC : 1.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

Figura 21. Conjunto de Emparejamiento para la Necesidad 'leer libro' Realizada por un Usuario con Enfoque Académico

Se puede observar en la Figura 21, que el nivel de enfoque del elemento de emparejamiento toma el valor de 0.5, esto debido a que el usuario con el que se realizó la consulta tiene un enfoque académico, al igual que el enfoque del dominio de librerías. Si se modifica la consulta de 'leer libro' por la de 'leer textos' o 'leer obras', se obtendrá el mismo conjunto de emparejamiento, con los mismos valores, debido a que dentro de las definiciones semánticas del elemento 'libro' se estableció que es semejante a 'textos' y 'obras'.

```
Información:   IMPRIMIENDO CONJUNTO DE EMPAREJAMIENTO
Información:   GRADO DE EMPAREJAMIENTO : 1.8
Información:   Nivel Entendimiento : 0.8
Información:   Nivel Enfoque : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:autor-libro ->
Información:   aso:recomendador-libro ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.7
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:servicio-tiquete ->
Información:   1/LC : 0.5
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   GRADO DE EMPAREJAMIENTO : 0.5333333333333333
Información:   Nivel Entendimiento : 0.2
Información:   Nivel Enfoque : 0.0
Información:   $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Información:   aso:comprador-cualidad ->
Información:   aso:producto-cualidad ->
```

Figura 22. Conjunto de Emparejamiento para la Necesidad 'buscar autor de Colombia'

La Figura 22 muestra el conjunto de emparejamiento obtenido para la necesidad 'buscar escritor Colombia' y 'buscar autor de Colombia'. El elemento con mayor grado de emparejamiento tiene el valor de 1.8, correspondiente a 0.8 de nivel de entendimiento de la consulta. La relación LC tiene el valor de 0.5, debido a que el camino necesidad-capacidad contiene dos asociaciones, tal como se ilustra en la Figura 23.

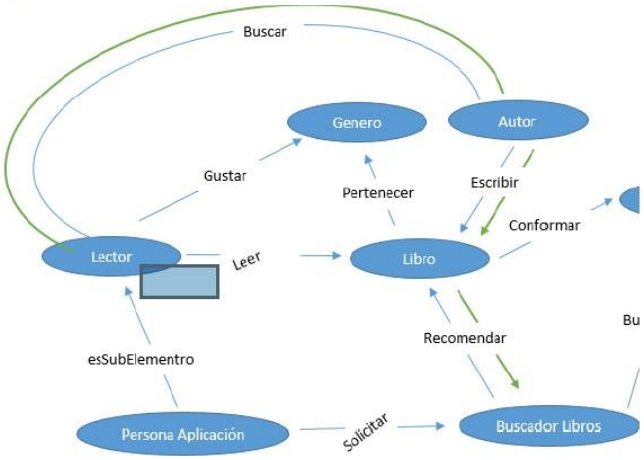

Figura 23. Camino Necesidad-Capacidad para la Necesidad 'buscar autor de Colombia'

Los resultados visualizados en la aplicación cliente emulada son mostrados en la Figura 24.

Novela	Gabriel José de la Concordia García Márquez	3	El libro narra la historia de la familia Buendia a lo largo de siete generaciones en el pueblo ficticio de Macondo	Cien años de soledad
Poesia	Rafael Pombo	4	poesia de simon el bobito. libro.	Simon el bobito

Figura 24. Resultados Obtenidos para la Necesidad 'buscar autor de Colombia'

VI. CONCLUSIONES

Con este trabajo se evidencia la viabilidad de realizar procesos de descubrimiento, composición y ejecución automáticos de SWS. En particular: a) El Framework FODAS-WS posibilita realizar especificaciones semánticas de SWS que le permiten a clientes solucionar necesidades específicas, b) Las necesidades expresadas por el cliente deben tener sintaxis acorde con las asociaciones de FODAS-WS, c) El MDCE-FODAS solo descubre, compone y ejecuta SWS especificados en FODAS-WS.

El presente trabajo usa el framework para la generación automática de aplicaciones a partir de la especificación semántica de procesos, el cual acude a la reutilización de servicios web semánticos ya implementados.

Se propone como trabajo futuro, la incorporación de tecnologías de representación de conocimiento, minería semántica y enlazado de datos, para mejorar la especificación semántica de los SW [10, 11]. Además, MDCE-FODAS debe ser extendido para considerar aspectos vinculados al Internet de las Cosas, Computación en la Nube, redes de sensores, entre otras cosas.

REFERENCIAS

[1] J. Aguilar, O. Portilla,"Framework Basado en ODA para la Descripción y Composición de Servicios Web Semánticos (FODAS-WS)", *Latin American Journal of Computing – LAJC*, Vol 2, No 2, pp. 15-24, 2015.

[2] S. Krishna, K. Jayashree, "Automatic Web Service Discovery Using Ontology Approach" International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 2 No 3, 2013

[3] E. Cho, S. Chung, "Automatic Web Services Generation", TCSS 702 Design Project in CSS (Winter 2008).

[4] K. Kiss, E. Blanco, Y. Cardinale, "ORINOCO Framework: Publicación, Composición y Ejecución de Servicios Web en Ambientes Grid". Novática: Revista de la Asociación de Técnicos de Informática, ISSN 0211-2124, págs. 40-47. 2010.

[5] W. PengWei, D. ZhiJun, J. ChangJun, "Automatic Web Service Composition Based on Uncertainty Execution Effects", IEEE Transactions on Services Computing, Vol 9, No 4, pp. 496 – 510, 2016

[6] P. Rodriguez, M. Mucientes, M. Lama, "Automatic Web Service Composition with a Heuristic-based Search Algorithm", IEEE International Conference on Web Services, 2011.

[7] A. Adala, N. Tabbane, S. Tabbane, " A Framework for Automatic Web Service Discovery Based on Semantics and NLP Techniques", Advances in Multimedia, 2011.

[8] Objetc Management Group, OMG (2003). MDA Guide Version 1.0.1 [online]. Massachusetts (USA): Object Management Group. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

[9] J. Timm, G. Gannod, "A Model-driven Approach for Specifying Semantic Web Services". 3rd. IEEE International Conference on Web Services, Vol. 1, pp. 313-320. 2015.

[10] J. Aguilar, J. Torres, K. Aguilar, "Autonomic Decision Making Based on Bayesian Networks and Ontologies", IEEE International Joint Conference on Neural Networks, pp. 3825-3832, 2016.

[11] C. Rangel, J. Altamiranda, J. Aguilar, "Semantic Mining Based on Graph Theory and Ontologies. Case Study: Cell Signaling Pathways", *CLEI Electronic Journal*, Vol. 19, no. 2, 2016.

[12] V. Bravo, J. Aguilar, F. Rivas, M. Cerrada, "Diseño de un Medio de Gestión de Servicios para Sistemas Multiagentes", XXX Conferencia Latinoamericana de Informática, pp. 431-439, 2004.

[13] J. Vizcarrondo, J. Aguilar, E. Exposito, A. Subias, "ARMISCOM: Autonomic Reflective MIddleware for Management Service Composition". 4th Global Information Infrastructure and Networking Symposium (GIIS), IEEE Communication Society, 2012.

[14] M. Sanchez, J. Aguilar, E. Exposito, "Fog Computing for the Integration of Agents and Web Services in an Autonomic Reflexive Middleware", Service Oriented Computing and Applications, Elsevier, 2018.