Plataforma para descubrimiento de servicios en ambientes ubicuos

Service discovery platform for ubiquitous environments

Víctor Alberto Hermida[*]*, Juan Carlos Corrales, Oscar Caicedo, Luis Suarez, Luis Rojas*

GIT, Universidad del Cauca, Calle 5ª N.° 4-70. Popayán, Colombia

(Recibido el 16 de octubre de 2009. Aceptado el 4 de noviembre de 2010)

Resumen

La capacidad de encontrar servicios útiles en un ambiente ubicuo se ha convertido en un problema crítico para diversos dominios de aplicación. Los enfoques actuales para la recuperación de servicios ubicuos están limitados en la mayoría de los casos al emparejamiento de sus entradas/salidas, posiblemente mejorado con un análisis semántico. Investigaciones recientes han demostrado que este tipo de aproximaciones no son suficientes para descubrir servicios relevantes. Por tanto, en el presente trabajo proponemos una plataforma de descubrimiento de servicios en ambientes ubicuos que tiene en cuenta la especificación del comportamiento de los componentes, de tal manera que es posible obtener un emparejamiento aproximado si no existe un servicio que corresponda exactamente con los requisitos del usuario.

----- *Palabras clave:* Ambientes ubicuos, recuperación de servicios, algoritmo

Abstract

The capability to easily find useful services in ubiquitous environments becomes increasingly critical in several application domains. Current approaches for ubiquitous services retrieval are mostly limited to the matching of their inputs/outputs possibly enhanced with some ontological knowledge. Recent research has demonstrated that this approach is not sufficient to discover relevant services. Motivated by these concerns, this work proposes a service discovery platform in ubiquitous environments based on behavior matchmaking. We argue that in many situations the service composition process requires a service discovery phase based on matchmaking of behavior specifications of the components. Consequently, even if a service that meets exactly the user requirements does not exist, the most similar ones will be retrieved.

----- *Keywords:* Ubiquitous environments, services retrieval, algorithm

* Autor de correspondencia: teléfono: + 57 + 2 + 820 98 00 ext 212, fax: + 57 + 2 + 820 98 13, correo electrónico: vhermida@unicauca.edu.co

Introducción

La computación ubicua puede ser descrita como dispositivos portables con facilidades embebidas de computación y comunicación, cuya meta es copar el ambiente con componentes electrónicos para ayudar de una manera natural a los usuarios durante la ejecución de sus tareas diarias [1]. En este sentido las principales características de la computación ubicua son: transparencia, movilidad y percepción del contexto [1]. Para lograr los objetivos de la computación ubicua se requiere abordar retos significativos, principalmente relacionados con la heterogeneidad del ambiente, el dinamismo y el contexto del usuario. La computación ubicua debe soportar las tareas de los clientes por medio de la integración de las funcionalidades de los servicios de red, de tal forma que éstos puedan ser invocados en cualquier momento y lugar [1]. Alcanzar tales objetivos es posible solo si se cuenta con mecanismos de recuperación de los servicios que están presentes en la red y que son requeridos por el usuario.

En un escenario típico de descubrimiento de servicios, el cliente especifica su requisito tomando en cuenta tres aspectos fundamentales [2]: i) que el servicio sea capaz de realizar una cierta función (p.ej. crear un carrito de compras), ii) exponer una interfaz en particular (p.ej. adicionar un producto), iii) proporcionar un comportamiento determinado (p.ej. ignorar cualquier petición de remover productos cuando el carrito de compras este vacío).

A pesar de lo expuesto, es muy difícil encontrar exactamente los servicios que busca un usuario, debido a que la mayoría de las veces se necesita adaptar los servicios disponibles en la red ubicua. En este sentido se plantea como interrogante: ¿Cómo soportar el descubrimiento de servicios en una ambiente de computación ubicua?. Para abordar la solución al interrogante planteado, el presente artículo expone una técnica de recuperación de servicios basada en modelos de comportamiento que permiten entregar una evaluación de la distancia semántica entre los servicios presentes en la red y los requisitos del usuario; inclusive si no se encuentra un servicio que responda exactamente a las peticiones del cliente, se recuperan los servicios con mayor similaridad realizando la búsqueda y emparejamiento de servicios a través de una representación formal de los servicios basada en grafos.

En la siguiente sección se presenta el estado actual del conocimiento, posteriormente se describe la plataforma propuesta para el descubrimiento y composición de servicios en ambientes ubicuos. La sección de metodología de evaluación detalla la forma como se determinó el desempeño del sistema para la recuperación de servicios, finalmente se presenta una discusión de los resultados obtenidos y las conclusiones.

Estado actual del conocimiento

El descubrimiento de servicios puede ser definido como la capacidad de encontrar y utilizar posteriormente un servicio basado en alguna descripción publicada de su funcionalidad y parámetros operacionales [3]. Las técnicas de emparejamiento pueden ser agrupadas en tres conjuntos: descubrimiento basado en interfaces, descubrimiento semántico y descubrimiento basado en comportamiento [4].

El descubrimiento basado en interfaces de servicio utiliza técnicas de comparación de las palabras clave que las describen (p.ej.: WSDL (*Web Services Description Language*), IDL (*Interface Definition Language*), RMI (*Remote Method Invocation*), etc.). Regularmente las técnicas de comparación de palabras clave requieren coincidencias exactas a nivel sintáctico entre las descripciones de los servicios, lo cual puede resultar en que se descarten servicios equivalentes a nivel lógico (p.ej. dos servicios descritos como Impresora e Impresión pueden ser distintos sintácticamente pero equivalentes lógicamente) [3,4].

La descripción semántica de los servicios proporciona un mayor nivel de razonamiento a los mecanismos de búsqueda e integración dinámica de los servicios. En [5 - 7] se presentan

técnicas de descubrimiento de servicios basadas en comparaciones semánticas, estos trabajos extienden la interfaz WSDL añadiendo atributos a las descripciones con el objetivo de soportar las comparaciones semánticas. El principal inconveniente de las anotaciones semánticas es la necesidad de una ontología común para las anotaciones y los servicios de consulta, la cual no existe en comunidades abiertas y heterogéneas como las redes ubicuas. La recuperación de servicios basada en comparaciones sintácticas o semánticas no es suficiente para un gran número de aplicaciones, trabajos recientes explotan el conocimiento sobre los servicios y su comportamiento. La necesidad de tomar en cuenta el comportamiento de un proceso fue resaltada por diversas investigaciones [8-10], quienes mejoran la precisión del descubrimiento de servicios web empleando los modelos de procesos para capturar el comportamiento de un servicio. En [11] se presenta una técnica de emparejamiento que opera sobre modelos de comportamiento, la cual permite medir la distancia entre los modelos almacenados y el modelo requerido por el usuario. Las técnicas son empleadas bajo dos protocolos BPEL (*Business Process Execution Language*) y WSCL (*Web Services Conversation Language*).

Teniendo en cuenta las diferentes aproximaciones presentadas, el objetivo del presente trabajo se centra en la recuperación de servicios en ambientes ubicuos a partir de modelos de procesos que expresan el comportamiento deseado por el usuario y que permiten entregar resultados aproximados basados en la distancia semántica de los servicios consultados.

Plataforma U-BeMatch

La plataforma U-BeMatch ofrece un servicio de descubrimiento y composición automático para ambientes de computación ubicua. El usuario interactúa con la plataforma por medio de una aplicación la cual soporta el envío de las peticiones expresadas como modelos de comportamiento BPEL. La plataforma recibe las peticiones y las transforma a una representación de grafos para ejecutar la recuperación de servicios. La recuperación de servicios es mejorada capturando el contexto de entrega del usuario. Posterior al descubrimiento se realiza un proceso de composición con los servicios recuperados para obtener un proceso similar al requerido por el usuario. Finalmente, la plataforma retorna el servicio compuesto para que los servicios sean invocados por medio de un motor BPEL. Por otro lado, los proveedores publican sus servicios como procesos BPEL, los cuales son almacenados en el repositorio de servicios y puestos a disposición para ser consumidos. La figura 1. Plataforma U-BeMacth ilustra la estructura de la plataforma, cada uno de los módulos es descrito a continuación.

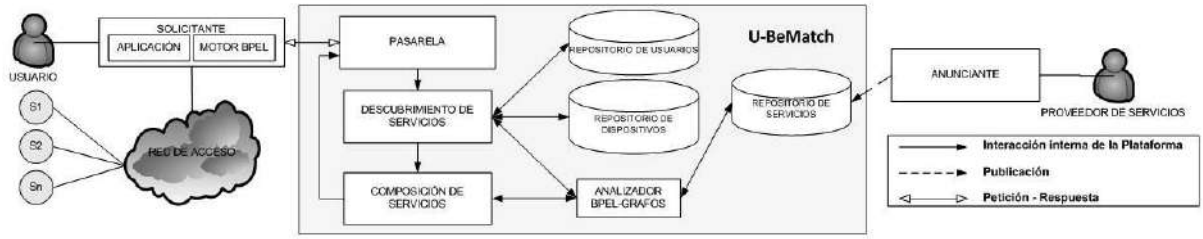

Figura 1 Plataforma U-BeMacth

Solicitante: módulo compuesto por una aplicación móvil y un motor BPEL. La aplicación ofrece la interfaz gráfica para la interacción con el usuario, además de proveer las capacidades de comunicación para el envío y recepción de los procesos BPEL. El motor BPEL es implementado por Sliver [12], un mediador que soporta la ejecución de procesos BPEL en un amplio rango de dispositivos, desde computadores de escritorio hasta dispositivos móviles.

Pasarela: recibe las peticiones enviadas desde el Solicitante y transmite las repuestas generadas por los niveles superiores. Este componente brinda un acceso independiente de la red (Wifi, Bluetooth, etc.) y de los protocolos (HTTP, SMTP, SMS, etc.) a la plataforma.

Anunciante: ofrece las capacidades necesarias para que los proveedores publiquen sus servicios en el repositorio de la plataforma. Las descripciones de servicios contienen atributos funcionales y no-funcionales, tales como: entradas, salidas, precondiciones, restricciones, etc. Este módulo es implementado a través de una aplicación web.

Repositorios: almacenan información sobre la descripción de los servicios disponibles y el contexto de entrega del usuario. El contexto de entrega es un conjunto de atributos que caracterizan las capacidades de acceso a la red ubicua, las preferencias del usuario y otros aspectos del contexto en el cual el servicio será consumido. A continuación se describen los repositorios utilizados: i) *Repositorio de servicios*: se utiliza el repositorio de procesos de negocio presentado en [13], el cual soporta el almacenamiento y consulta de documentos BPEL (y otros documentos XML). ii) *Repositorio de usuarios*: almacena detalles relacionados con los usuarios, estos incluyen preferencias explicitas provistas por los usuarios (ej. Lenguaje nativo) o datos implícitos capturados dinámicamente tales como historial de servicios consumidos o patrones de uso. iii) *Repositorio de dispositivos*: almacena las características de los dispositivos tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. Esta información se recupera de los repositorios: UAProf (User Agent Profile) [14] y WURFL (Wireless Universal Resource) [15].

*Analizador BPEL-Grafos***:** transforma las descripciones de comportamiento (BPEL) en su equivalente en grafos implementando la estrategia presentada por [16]. El algoritmo emplea un proceso de transformación recursivo para cada tipo de actividad estructurada tomando una aproximación de arriba-abajo (top-down). Las actividades básicas BPEL son transformadas en nodos y las secuencias son obtenidas conectando los nodos requeridos por medio de aristas. Las actividades estructuradas son representadas por medio de operadores lógicos XOR y AND.

Módulo de descubrimiento de servicios: recupera los servicios más apropiados para componer el modelo de comportamiento requerido por el usuario. El proceso de descubrimiento se detalla en el algoritmo 1.

Algoritmo 1 Algoritmo de descubrimiento

INPUTS: BPELDocument
OUTPUT: *List Activities*

List Activity activities*PreviouslyConsumed = searchActivitiesConsumed (BPELDocument)*
if activities*PreviosulyConsumed* is not null (required activities have already been consumed by the client) **then Return** *activitiesPreviouslyConsumed*
else
ListActivity *candidateActivities = lookupServiceRepository(non-operational information)*
Struct(nodes) *queryGraph = parserBPELToGraph(BPELDocument)*
List Activity *rankedActivities = matchingActivities(queryGraph, candidateActivity)*
List Activity rankedFilteredActivities = checkDeliverycontext(rankedActivities)
Return *rankedFilteredActivities*
end if

El proceso de descubrimiento tiene como entrada el documento BPEL enviado por el cliente y entrega una lista de actividades candidatas. El descubrimiento inicia en el Repositorio de usuario consultando las actividades previamente consumidas por el cliente (*searchActivitiesConsumed*), si las actividades requeridas ya han sido consumidas por el usuario la plataforma obtiene la referencia de los servicios respectivos consultando el Repositorio de usuario, en caso contrario se avanza a la etapa de emparejamiento. A continuación, se realiza el primer paso seleccionando los posibles servicios candidatos basándose en información no-operacional, por ejemplo realizando un filtrado de los servicios utilizando dominios o áreas de cubrimiento del proveedor (*lookupServiceRepository*). Terminado el primer paso se tiene una lista de actividades candidatas que serán comparadas con las actividades requeridas por el cliente. El documento BPEL de consulta es transformado a su representación en grafos (*parserBPELToGraph*), obteniendo una lista de nodos que representan las actividades básicas que requiere el usuario para ejecutar el modelo de comportamiento deseado. En este momento se realiza la comparación de las actividades requeridas por el usuario y las actividades publicadas en los repositorios por medio de la función matching*Activities*. La función *matchingActivities* compara las actividades básicas de entrada con las contenidas en el repositorio, esta comparación se basa en la función *BasicAvtivityMatch*, la cual se detalla en la siguiente sección. El último paso consiste en verificar si las actividades recuperadas pueden ser invocadas en el contexto del usuario (*checkDeliverycontext*).

Emparejamiento de Actividades: al emplear una representación de grafos para los requerimientos de usuarios y los servicios almacenados en el repositorio, el problema de emparejamiento de actividades se transforma en un problema de emparejamiento de nodos. El emparejamiento de actividades se realiza a nivel atómico, comparando las actividades básicas que conforman el requisito del cliente contra las actividades contenidas en los repositorios. La función BasicActivityMatch (ver Algoritmo 2) toma como entradas dos nodos, que representan actividades básicas de BPEL (receive, invoke, reply), y calcula la distancia semántica entre los dos. Cada nodo posee dos atributos Nombre de la operación (Op) y el PortType (PT). La función de emparejamiento le da prioridad a la comparación de la operación, si las dos operaciones son similares (SimOperation > 0) se calcula la similaridad del PortType y se estima la distancia entre las dos actividades (DistanceNode). Los pesos *Wop* y *Wpt* indican la contribución de la similaridad de operación y PortType a la similaridad de las actividades ($0 \leq Wop \leq 1$ y $0 \leq Wpt \leq 1$). Para calcular la similaridad de los atributos se emplea la función *LS*, implementada por el Analizador Lingüístico.

Algoritmo 2 Algoritmo de BasicActivityMatch

INPUTS: (Nodei, Nodej) Nodei: Struct (Opi, PTi), Nodej: Struct (Opj, PTj)
OUTPUT: *DistanceNode*

Calculate Operation Similarity *SimOperation = LS(Opi, Opj)*
if SimOperation = 0 (different Operations) **then Return** *DistanceNode* = 1
else
Calculate Port Type Similarity *SimPortType = LS(PTi, PTj)*
Calculate *DistanceNode* $\text{DistanceNode} = 1 - \frac{w_{op} * \text{SimOperation} + w_{pt} * \text{SimPortType}}{w_{op} + w_{pt}}$
end if

Analizador lingüístico: el módulo de comparación lingüística calcula la similaridad lingüística entre dos etiquetas basándose en sus nombres. Para obtener esta medida se utilizan los algoritmos *Ngram, Check synonym* y *Check abbreviation.* El algoritmo *Ngram* estima la similaridad de acuerdo al número común de *qgramas* entre las etiquetas [17]. El algoritmo *Check synonym* utiliza el diccionario lingüístico Wordnet [18], para identificar sinónimos, mientras el algoritmo C*heck abbreviation* usa un diccionario de abreviaciones adecuado al dominio de aplicación. Si bien es cierto que el presente trabajo utiliza el diccionario Wordnet, la plataforma U-bematch puede hacer uso de cualquier otro tipo de diccionario. Si todos los algoritmos entregan un valor de 1 existe una coincidencia exacta entre las etiquetas, si entregan un valor de 0 no hay similaridad entre las palabras. Si los valores entregados por *Ngram* y *Check abbreviation* son iguales a 0 y el valor de *Check synonym* está entre 0 y 1, el valor total de la similaridad es igual a *Check synonym.* Finalmente, si los tres algoritmos arrojan un valor entre 0 y 1, la similaridad lingüística es el promedio de los tres. La función LS es descrita en el algoritmo 3.

Algoritmo 3 Algoritmo Función LS

$$LS = \begin{cases} 1, & si\ (m1 = 1\ \vee m2 = 1\ \vee m3 = 1) \\ m2, & si\ (\ 0 < m2 < 1\ \wedge\ m1 = m3 = 0) \\ 0, & si(m1 = m2 = m3 = 0) \\ (m1 + m2 + m3)/3, & si\big(m1, m2, m3\ \in (0,1)\big) \end{cases}$$

$$Donde: \boldsymbol{m1} = Sim(NGRAM);\ \boldsymbol{m2} = Sim(SynonymMatching);\ \boldsymbol{m3} = Sim(AbbreviationExpansion)$$

Módulo de composición de servicios: se encarga de componer el proceso más similar al servicio requerido por el cliente, usando los servicios recuperados previamente. La composición se realiza teniendo en cuenta el flujo de control definido en la petición del usuario. En esta fase se genera un grafo que emplea las actividades recuperadas como nodos y define sus aristas a partir del grafo de consulta. Finalmente el grafo generado es transformado a un proceso de negocio BPEL para ser entregado al usuario.

Metodología de evaluación

Para medir el desempeño del algoritmo de recuperación de servicios se procedió de la siguiente manera: i) Se construyó un banco de pruebas para comparar un conjunto de actividades de consulta contra un repositorio de servicios, este banco de pruebas constituye una verdad absoluta sobre la similaridad de las actividades comparadas. ii) Se compararon los resultados arrojados por el algoritmo de recuperación contra el banco de pruebas, con el objeto de medir su desempeño, para ello se definieron previamente los criterios para cuantificar el rendimiento del sistema.

Banco de pruebas: Un banco de pruebas puede definirse como un proceso sistemático y continuo para evaluar comparativamente los productos, servicios y procesos de trabajo en organizaciones [19]. Con el propósito de tener una base para comparar los resultados del algoritmo de búsqueda de actividades de procesos de negocios se estableció un banco de pruebas conformado a partir de 30 actividades básicas BPEL agrupadas en 5 dominios: Vacaciones (5 actividades), Compras (7), Pagos (4), Disponibilidad de productos (7) e Información de productos (7). El banco de pruebas se creó comparando las 30 actividades básicas BPEL contra 144 actividades almacenadas en el repositorio de servicios. Las comparaciones fueron realizadas por 5 evaluadores, obteniendo 5530 comparaciones por cada uno, para un total

de 27.650 comparaciones. Las comparaciones realizadas para las actividades BPEL evalúan la similaridad de cinco atributos: Activity Type, Operation, Portype, PartnerLink y Access Type (ver figura 2), los cuatro primeros atributos se usan para describir una actividad básica en un documento BPEL y son empleados por el algoritmo de comparación explicado en la sección anterior. El atributo Access Type corresponde a una extensión que se realiza a la descripción de los servicios, el cual será empleado cuando se implementen las funciones de verificación del contexto de entrega. El evaluador realizó la comparación entre las actividades asignando una calificación a cada uno de los atributos según su similaridad. El valor de la calificación está entre 0 y 5, 0 mínima similaridad y 5 máxima similaridad. El experto evaluador fija un peso de acuerdo a la importancia de cada uno de los atributos, la suma total de los pesos debe ser igual a 1.

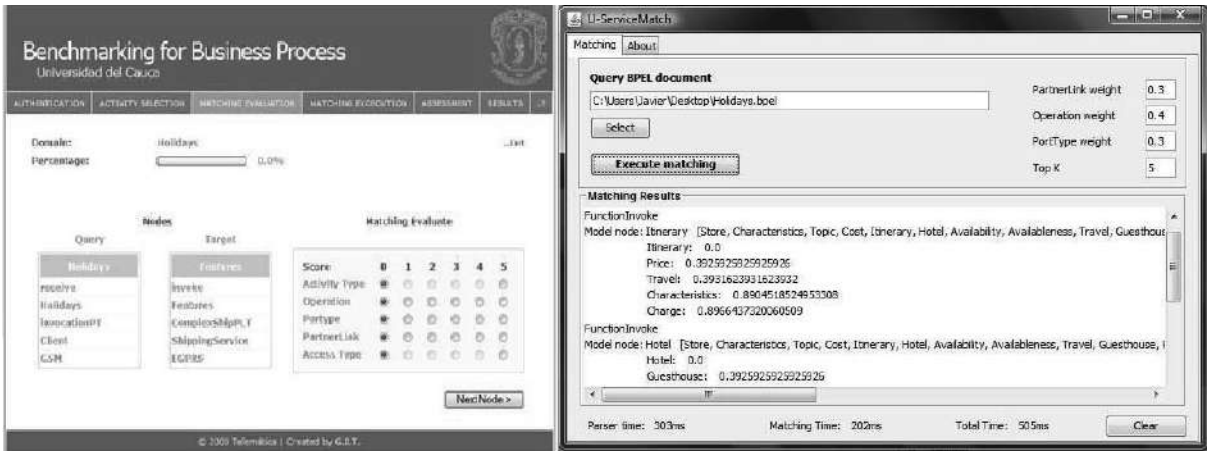

Figura 2 Evaluación de similaridad de actividades

El valor de similaridad para cada comparación es calculado de la siguiente manera:

1. Similaridad: $EMu = \Sigma_{i=1}^{5} (Wi * Sui)$;

   Donde: *Wi* (pesos) y *Sui* (Calificación)

El valor *EMu* mide la similaridad estimada por el usuario para un par de actividades. La media de similaridad para cada una de las actividades evaluadas es la siguiente:

2. Media de Similaridad: $TE(EM) = \frac{\Sigma_{u=1}^{n} EMu}{n}$

   Donde *EMu* = similaridad de una comparación; *n* = número de evaluadores

Para cada una de las 30 actividades evaluadas en el banco de pruebas se obtiene una lista ordenada conformada por las 144 actividades del repositorio con su respectivo valor de similaridad media (TE(Em)), identificando las actividades relevantes que deben ser recuperadas del repositorio. De esta forma se obtiene el banco de datos que permitirá realizar las medidas de recuperación del algoritmo de emparejamiento.

Medidas de Desempeño: El desempeño general del sistema se ha establecido utilizando las medidas *recall (r), precision (p), total (o), top-k precisión* (p_k) y *p-presicion* (p_p). Para evaluar la calidad del algoritmo de recuperación, se comparan las actividades ***(P)*** retornadas por el algoritmo con las actividades ***(R)*** obtenidas en el *banco de pruebas.* De esta forma se puede determinar un conjunto de *verdaderos positivos* (***I***), actividades correctamente identificadas; igualmente se determina un conjunto de *falsos positivos*, actividades falsas recuperadas *(F = P/I),* y *falsos negativos*, es decir actividades relevantes no recuperadas *(M = R/I)* [4]. $Retrel_k$ es el conjunto de actividades relevantes para un top k de actividades recuperadas, mientras *Rel-p* determina cuantas de las actividades de

$Retrel_k$ están en la misma posición del ranking de referencia del banco de pruebas [20]. Con base en la cardinalidad de estos conjuntos se tiene:

$$\boldsymbol{p} = \frac{|\boldsymbol{I}|}{|\boldsymbol{P}|} \ , \ \boldsymbol{r} = \frac{|\boldsymbol{I}|}{|\boldsymbol{R}|} \ ,$$

$$\boldsymbol{o} = \boldsymbol{r} * \left(\mathbf{2} - \frac{\mathbf{1}}{\boldsymbol{p}}\right),$$

$$\boldsymbol{p_k} = \frac{|\boldsymbol{Retrel_k}|}{\boldsymbol{k}}, \ \boldsymbol{p_p} = \frac{[\boldsymbol{Retrel_k}]\boldsymbol{Rel-p}}{\boldsymbol{k}}$$

La medida *precision* estima la fiabilidad de los servicios relevantes recuperados por el algoritmo, en tanto que *recall* especifica el porcentaje de los servicios relevantes recuperados. Por su parte la medida *total* valora la calidad del emparejamiento, teniendo en cuenta el esfuerzo necesario para la eliminación de falsos positivos y los servicios no recuperados.

Las medidas establecidas anteriormente se calculan para cada una de las actividades BPEL empleadas en el banco de pruebas. Para estimar la *precision* y *recall* de todo el sistema, se emplean los métodos macro-promedio y micro-promedio [21], así:

Macro-promedio: es la media de la *precisión* y *recall* de los emparejamientos individuales.

$$P = \frac{\sum_{i=1}^{n} p_i}{n} \ , \ R = \frac{\sum_{i=1}^{n} r_i}{n}$$

Donde ***n****:* es el número de emparejamientos realizados

Micro-promedio: tiene en cuenta los verdaderos positivos y los falsos positivos. La precisión y el recall se calculan utilizando los valores globales.

$$P = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FP_i)} \ , \ R = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FN_i)}$$

Donde: ***TPi***= verdaderos positivo; ***FPi=*** falsos positivo*;* ***FNi***= falsos negativos.

Resultados y discusión

Se implementó un sistema de recuperación de servicios sobre un repositorio de documentos BPEL [13], el cual almacena 17 archivos BPEL y cuenta con un total de 144 actividades básicas BPEL. El algoritmo de emparejamiento se ha implementado en el lenguaje Java y los experimentos fueron realizados en un computador con procesador Pentium 4 2,30GHz, 1.000 MB de RAM bajo el S.O. Linux Ubuntu.

Se evaluaron las medidas de *precision, recall, total, Top-k precision, P-Precision K* tomando como referencia un banco de pruebas de 30 actividades, para las cuales se obtuvieron las actividades más similares de acuerdo al algoritmo de descubrimiento. Los valores globales de las medidas se calcularon por medio de las técnicas de *Macro-promedio* y *Micro-promedio*.

Para el análisis de los resultados se plantean tres escenarios: i) evaluación de las medidas de desempeño comparando las actividades de entrada contra las actividades del repositorio que pertenecen al mismo dominio (caso 1), ii) evaluación comparando las actividades de entrada contra las actividades almacenadas que pertenecen a un dominio diferente (caso 2), y iii) evaluación comparando las actividades de consulta contra todas las actividades contenidas en el repositorio (caso 3). La figura 3 Medidas de desempeño ilustra los resultados obtenidos para los dominios utilizados en el banco de pruebas.

En la figura 3(a) se puede observar la relación entra las medidas de *precision* y *recall* para los tres escenarios de prueba. En el caso 1 se tiene una mejor relación entre la medida de *precision* y *recall*, esto se puede atribuir al hecho que se comparan actividades de un mismo dominio, teniendo el sistema una mayor posibilidad de diferenciar entre las actividades relevantes y las que se deben descartar. Por su parte el caso 3 toma el mismo comportamiento que el caso 1, sin embargo su eficacia es disminuida por el ruido que agregan las actividades pertenecientes a otro dominios. El caso 2, presenta un comportamiento

particular al ser muy preciso, ya que las actividades recuperadas son todas relevantes para la búsqueda, sin embargo esta precisión implica que se descarten muchas actividades que podrían ser candidatas relevantes en la búsqueda, hecho por el cual el valor de *recall* es bajo y no crece más allá de un 0,2. El desempeño del caso 2 se presenta como el más pobre de los tres escenarios, ya que se tiene una gran diferencia entre los valores de *precisión* y *recall*.

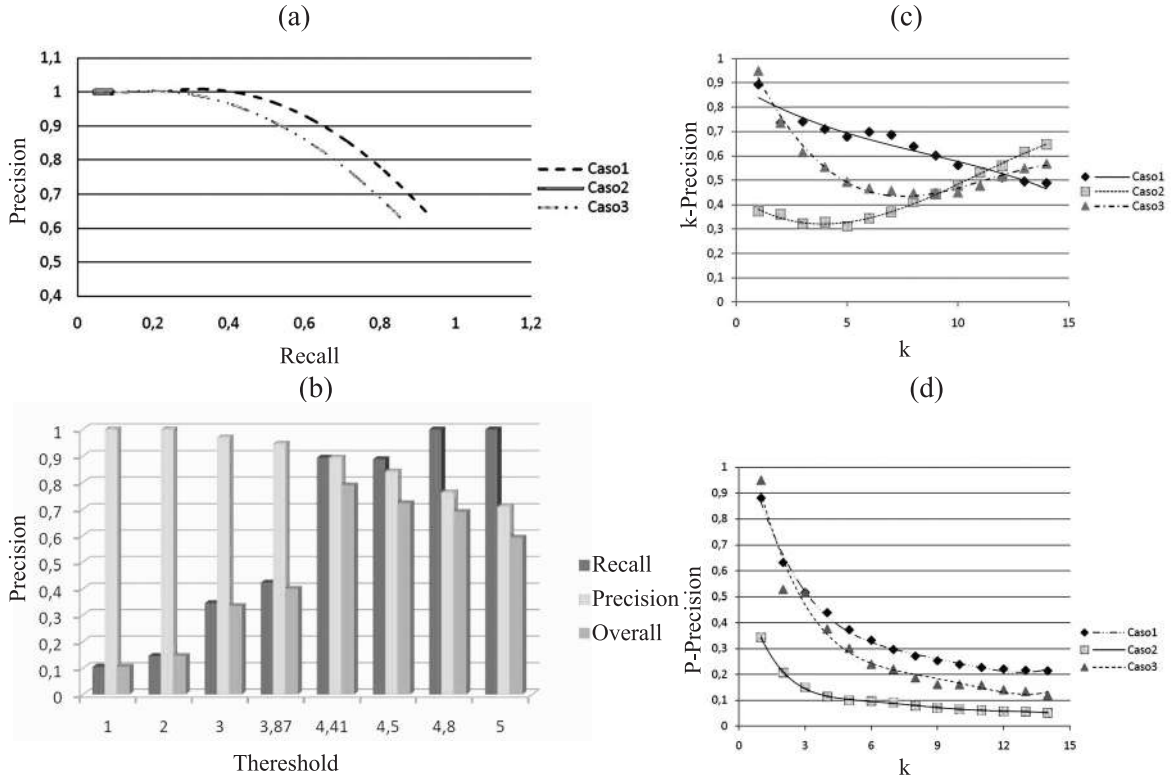

Figura 3 Medidas de desempeño

El desempeño total del sistema se puede apreciar en la figura 3(b) donde se identifica el umbral de similaridad óptimo para el algoritmo de emparejamiento, equivalente a 4,41, punto en cual la medida *total* alcanza su tope en un valor cercano al 80%. Igualmente se aprecia que en los valores de similaridad superiores a 4,41 el sistema obtiene un mejor desempeño que aquellos ubicados en la parte inferior, siendo más favorable trabajar con valores altos de similaridad, donde la medida de *recall* realiza un mayor aporte al desempeño general. El algoritmo trabaja de una manera muy selectiva, tomando valores de precisión muy altos, y recuperando un alto porcentaje de servicios relevantes cuando el umbral de similaridad está por encima de 4.

La medida de precisión con respecto al número de actividades recuperadas por el algoritmo se presenta en la figura 3(c). Para valores bajos de k (número de actividades recuperadas) se tiene que el caso 1 es más preciso, dado que la búsqueda bajo el mismo dominio de consulta incrementa la posibilidad de encontrar las actividades más similares a una actividad requerida. Por el contrario el caso 3, inicia con valores pequeños de precisión para un k bajo, pero su precisión crece a medida que se aumenta el número de actividades recuperadas, hecho atribuido a la posibilidad de obtener actividades con un valor menor de similaridad. De acuerdo a la medida *Total* el mejor desempeño del sistema se obtiene para valores de similaridad superiores a 4,41,

rango en el cual la precisión oscila entre 0,7 y 0,9. Tomando este rango como referencia el k óptimo del sistema se encuentra entre 1 y 7 actividades recuperadas, valores en los que la precisión se ajusta a los parámetros requeridos para el mejor desempeño. Para alcanzar estos valores se debe implementar un filtrado de actividades por dominio que ayude a obtener un comportamiento similar al caso 1 de la evaluación realizada.

La gráfica de P-Precisión muestra de igual manera que el caso 1 acierta con una mayor eficacia la posición y las actividades recuperadas de acuerdo al banco de pruebas generado. Para esta medida las curvas de la figura 3(d) son decrecientes a medida que se incrementa el número de actividades k, comportamiento presentado en los tres casos. Las mejores presiones de posición se alcanzan para un valor de k inferior a 3, teniendo como valor mínimo de precisión 0,7 rango en el cual se tiene un desempeño total óptimo.

Conclusiones

El enfoque presentado usa técnicas de emparejamiento que trabajan sobre modelos de comportamiento. Se realiza una evaluación de la distancia semántica entre las actividades requeridas por el usuario y aquellas contenidas en el repositorio de la plataforma, soportando la entrega de aproximaciones cuando no se obtiene una coincidencia exacta de los parámetros de búsqueda. El problema del emparejamiento de actividades es abordado por medio de una representación de grafos que permite extraer las actividades básicas requeridas y obtener un formalismo para procesos posteriores de composición de las actividades recuperadas.

El desempeño del algoritmo de descubrimiento de servicios se midió tomando como referencia un banco de pruebas construido para este propósito. Se determino un umbral de similaridad óptimo equivalente a 4,41, valor en el cual se alcanza el máximo desempeño del sistema de recuperación, se evidenció que el desempeño es mejor cuando se emplean valores de similaridad superiores al umbral, ya que en valores bajos de similaridad la medida de *recall* es muy pobre al descartar demasiadas actividades consideradas como relevantes. A partir de este umbral se concluye que el número *k* de actividades debe estar entre 1 y 7, rango en el que los valores de *precisión* son adecuados para un desempeño óptimo. Este rango de valores para el *k* y la similaridad permiten establecer los parámetros para obtener las actividades más relevantes para el usuario teniendo en cuenta el documento BPEL de consulta.

Como trabajo futuro se adicionarán los repositorios para determinar el contexto de entrega, además se construirá un módulo de composición para generar los servicios que respondan a los requerimientos de los usuarios, para esto se necesitará encontrar un nuevo umbral óptimo de similaridad tomando en cuenta la información aportada por los nuevos repositorios.

Referencias

1. E. S. Abdur-Rahman, J. Black. "Semantic-based context-aware service discovery in pervasive-computing environments". *Proceedings of 1st IEEE Workshop on Service Integration in Pervasive Environments*. Lyon. France. 2006. pp. 9-14.

2. J. C. Corrales, D. Grigori, M. Bouzeghoub. "BPEL Processes Matchmaking for Service Discovery". M. Papazoglou, L. Raschid, R. Ruggaber (editors) *Proceedings of the 4th International Conference on Cooperative Information Systems (CoopIS 2006)*. Montpellier. France. October 29 - November 3. 2006. pp. 237-254.

3. A. Bandara, T. Payne, D. De Roure, T. Lewis. *A Semantic Approach for Service Matching in Pervasive Environments*. Reporte Técnico. Universidad de Southampton. Southampton (UK). 2007. pp 1-5.

4. J. C. Corrales. *Behavioral matchmaking for service retrieval*. Ph.D.Tesis. University of Versailles Saint-Quentin-en-Yvelines. Versailles (France). 2008. pp 26-44.

5. M. Sellami, S. Tata, B. Defude. *Service Discovery in Ubiquitous Environments: Approaches and Requirement for Context-Awareness. Advances in Semantics for Web services Workshop*. Ed. *BPM Workshops*. Vol 17. 2008. Milan (Italy). 2008. pp. 516-522.

6. L. Steller, S. Krishnaswamy. "Efficient Mobile Reasoning for Pervasive Discovery". *Proc of the 2009 ACM symposium on Applied Computing*. Honolulu (Hawaii). March 9-12. 2009. pp. 1247-1251.

7. Y. Zhang, B. Liu, H. Wang. "A Method of Web Service Discovery based on Semantic Message Bipartite Matching for Remote Medical System". *Journal of Theoretical and Applied Electronic Commerce Research*. Vol. 4. 2009. pp. 79-87.

8. G. H. T. Sayed, M. Wan, I. Suhaimi, V.D. Amir. "Integrating Discovery and Composition of Semantic Web Services Based on Description Logic*". Journal of Computer Science, Informatics & Electrical Engineering*. Vol. 3. 2009. pp. 1-12.

9. A. Bernstein, M. Klein. "Towards high-precision service retrieval". *Proceedings of ISWC, LNCS*. Sardinia (Italia). June 9-12. 2002. Vol. 2342. 2002. pp. 84-101.

10. A. Wombacher, B. Mahleko, P. Fankhauser, E. Neuhold. "Matchmaking for business processes based on choreographies". *Proceedings of IEEE Computer Society.* February 19-20.2004. Washington DC. (USA). 2004. pp. 359-368.

11. J. C. Corrales, D. Grigori, M. Bouzeghoub, J. E. Burbano. "Bematch: A platform for matchmaking service behavior models". *Proceedings of EDBT. ACM International Conference Proceeding.* Series 261. Nantes (France). 2008. pp. 695-699.

12. G. Hackmann, C. Gill, G. C. Roman. "Extending BPEL for interoperable pervasive computing". *Proceedings. of ICPS. IEEE Computer Society Press*. Istanbul (Turkey). 2007. pp. 204-213.

13. J. Vanhatalo, J. Koehler, F. Leymann. "Repository for business processes and arbitrary associated metadata". *Proceedings of the BPM Demo Session at the Fourth International Conference on Business Process Management*. Viena (Austria). 2006. pp. 25-31.

14. Wireless Application Protocol Forum, WAG UAProf version 20-Oct-2001. http://www.wapforum.org/. 2001. Consultada el 7 de octubre de 2009.

15. L. Passani, http: //wurfl.sourceforge.net/. Consultada el 7 de octubre de 2009.

16. J. Mendling, J. Ziemann. "Transformation of bpel processes to epcs". *Proceedings of EPK*. Hamburg (Germany). Vol. 167. 2005. pp. 41-53.

17. R. C. Angell, G. Freund, P. Willett. "Automatic spelling correction using a trigram similarity measure". *Information Processing and management*. Vol. 19. 1983. pp. 255-261.

18. G. Miller. "Wordnet: A lexical database for English". *Communications of the ACM*. Vol. 38. 1995. pp. 39-41.

19. M. J. Spendolini. *Benchmarking*. Ed. Amacon. New York (USA). 1994. pp. 3-50.

20. Y. Zhang, X. Dong, A. Halevy, J. Madhavan, E. Nemes. "Similarity Search for Web Services". *Proceedings of the 30th VLDB conference*. Toronto (Canada).Vol. 30. 2004. pp. 372–383.

21. D. Lewis. *Representation and learning in information retrieval*. Ph.D. Thesis. Department of Computer and Information Science. University of Massachusetts. Massachusetts (USA). 1992. pp. 28-35.