Descubrimiento de Servicios en Ambientes Ubicuos

Universidad del Cauca

Monografía presentada para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Luis Javier Suarez Meza
Luis Antonio Rojas Potosí

Director: PhD. Ing. Juan Carlos Corrales

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación Servicios Avanzados de Telecomunicaciones
Popayán, Septiembre de 2010

Tabla de contenido

Capítulo I 1
1.1. Contexto 1
1.2. Motivación 3
1.3. Planteamiento del Problema 3
1.4. Objetivos 4
1.5. Contribuciones y Principales Resultados. 4
1.6. Contenido de la Monografía. 6
Capítulo II 9
2.1. Conceptos Generales 9
2.1.1. Computación Ubicua 9
2.1.2. SOA (Service Oriented Architecture) 10
2.1.3. Composición de Servicios en Ambientes de Computación Ubicua 12
2.1.4. Descubrimiento de servicios 13
2.1.5. Servicios Web 14
2.1.6. Procesos de negocio 15
2.1.7. Estándares para la ejecución de procesos 15
2.3. Trabajos relacionados 19
2.2.1. Descubrimiento de Servicios en ambientes Ubicuos 19
2.2.2. Personalización 22
2.4. Resumen 26
Capítulo III 29
3.1. Emparejamiento 29
3.1.1. Modelos Formales para Representación de Procesos 30
3.1.2. Transformación de BPEL a Grafos 31
3.1.3. Emparejamiento de Actividades Básicas 34
3.1.4. Ejemplo de Matching de Actividades Básicas. 36
3.2. Personalización 38
3.1.5. Gestión del Contexto 39
3.1.6. Perfil de Usuario 41
3.1.7. Ejemplo de personalización 44
3.3. Resumen 46

Capítulo IV 47
4.1. Establecimiento de responsabilidades 47
4.1.1. Consideraciones Iniciales 48
4.1.2. Captura y Análisis de Requisitos Funcionales 49
4.1.3. Identificación de Casos de Uso 51
4.2. Descripción de la solución 53
4.2.1. Diagramas de clases del Sistema 53
4.2.2. Diagramas de paquetes del Sistema 59
4.3. Definición de la Arquitectura 65
4.3.1. Diagrama de Despliegue de Plataforma U-ServiceMatch 67
4.4. Resumen 69
Capítulo V 71
5.1. Metodología de Evaluación 71
5.1.1. Benchmarking 71
5.2. Plataforma de Evaluación Be4SeD 72
5.2.1. Banco de Servicios 73
5.2.2. Benchmark de Referencia 73
5.2.3. Políticas de Creación del Benchmark de Referencia 73
5.2.4. Benchmark del Algoritmo 73
5.2.5. Análisis estadístico 74
5.3. Prototipo de la Plataforma Be4SeD 75
5.3.1. Banco de Servicios 75
5.3.2. Conversor BPEL-Grafos 76
5.3.3. Control de Información 76
5.3.4. BenchMarkingDB 76
5.3.5. Interfaz de Usuario 76
5.3.6. Módulo de Ejecución 78
5.3.7. Módulo de Recolección de Información 78
5.3.8. Módulo de Análisis 78
5.4. Criterios de Evaluación 79
5.5. Plan de Pruebas 80
5.6. Especificaciones Técnicas del Servidor Central y de los Dispositivos móviles usados en las pruebas. 81

5.7. Resultados 81
5.8. Resumen 90
Capítulo VI 93
6.1. Contribuciones 93
6.2. Conclusiones 94
6.3. Trabajos futuros 96
Referencias 99

Anexos

Anexo A Artefactos Obtenidos en la Planeación, Construcción y Elaboración de la Plataforma U-ServiceMatch

Anexo B Repositorios

Anexo C Manual de Instalación y de Usuario del Repositorio de procesos BPEL

Anexo D Tablas de Resultados de las Pruebas de Calidad y Rendimiento del Sistema

Anexo E Artículos de Publicación

Índice de Ilustraciones

Figura 1. Las tres eras de la Computación. (Weiser, 96) 1
Figura 2. Elementos Conceptuales de SOA 11
Figura 3. Elementos Conceptuales de SOA 12
Figura 4. Plataforma U-BeMatch 13
Figura 5. Elementos principales de un proceso BPEL 17
Figura 6. Modelo conceptual básico 29
Figura 7. Correspondencia entre elementos BPEL y elementos de Grafos 34
Figura 8. Ejemplo de Emparejamiento de Actividades Básicas BPEL 37
Figura 9. Meta-Dato de los requerimientos de Contexto del Servicio. 39
Figura 10. Meta-Modelo de Contexto del Usuario 40
Figura 11. Meta-modelo del perfil de usuario 42
Figura 12. Diagrama de Casos de Uso del Sistema 53
Figura 13. Diagrama de Clases aplicación ServiceMatch 55
Figura 14. Diagrama de Clases aplicación SeMatch-Context 58

Figura 15. Diagrama de Clases del Repositorio de Usuarios 59
Figura 16. Vista lógica de la Aplicación ServiceMatch.....60
Figura 17. Vista lógica de la Aplicación SeMatch-Context.....62
Figura 18. Diagrama de Paquetes U-ServiceMatch.....65
Figura 19. Arquitectura de la Plataforma U-ServiceMatch 66
Figura 20. Diagrama de Despliegue de la Plataforma U-ServiceMatch 67
Figura 21. Arquitectura genérica de Be4SeD 72
Figura 22. Diagrama de Despliegue de la plataforma Be4SeD 76
Figura 23. Interfaz de Selección de Actividades a evaluar. 77
Figura 24.Interfaz de Evaluación. 77
Figura 25. Ranking de Servicios.....79
Figura 26. Rendimiento del proceso de Transformación de los documentos BPEL del Repositorio de Servicios en Grafos, además de obtener, clasificar y organizar los nodos por tipo de actividad.....82
Figura 27. K-Precisión vs. K.....83
Figura 28. P-Precisión vs. K.....84
Figura 29. Precisión vs. Recall.....84
Figura 30. Umbral del Sistema.....85
Figura 31. Rendimiento del proceso de Matching de las actividades básicas BPEL.....86
Figura 32. . Medidas de Desempeño para la aplicación SeMatch-Context.....87
Figura 33. Rendimiento del proceso de Matching de las actividades básicas BPEL considerando el contexto. 88
Figura 34. Gráfica de rendimiento en el proceso de creación del contexto de entrega. 89
Figura 35. Gráfica de rendimiento en el proceso de recuperación de servicios sobre diferentes terminales móviles para la aplicación SeMatch-Context. 89

Índice de tablas

Tabla 1. Evaluación de Modelos Formales de Representación de Procesos..... 30
Tabla 2. Plan de Pruebas..... 81
Tabla 3. Especificaciones técnicas del Servidor Central de U-ServiceMatch..... 81
Tabla 4. Especificaciones técnicas de los dispositivos usados para las pruebas. 81
Tabla 5. Archivos Query BPEL..... 86
Tabla 6. Formulas linealizadas de rendimiento 91

Capítulo I

INTRODUCCIÓN

1.1. Contexto

Internet es una de las herramientas científicas y tecnológicas más grandes e importantes desarrollada por el hombre, es una infraestructura omnipresente de información que proporciona datos, comunicación de voz e imágenes, entre otras funcionalidades, para miles de millones de personas en todo el mundo. El siguiente paso en su vertiginoso avance es la integración de elementos físicos a esta infraestructura de información, permitiendo la creación de nuevos y mejores servicios. De ésta forma, la siguiente generación de comunicaciones móviles será diferente a los sistemas actuales. Se prevén cambios en el tipo de acceso y en los dispositivos empleados para interactuar con las redes. Se considera el empleo de terminales distribuidos, configurables dinámicamente y en proximidades a los usuarios que permitirán un acceso permanente a la información, abriendo paso a la visión introducida por Weiser con el nombre de Computación Ubicua (Weiser, 1997). Según Weiser, la computación ubicua se describe como la existencia de pequeños ordenadores, con capacidades de comunicación y computación, embebidos de forma casi invisible en cualquier tipo de dispositivo cotidiano, integrándose amigablemente con los humanos. Es decir, las personas interactúan con ellos de forma inconsciente (Almenárez, 2005)(Weiser, 1991).

Weiser describe tres olas en computación, Figura 1: la primera denominada la era del computador, en la que muchas personas compartían un computador; la era del computador personal, en la que cada persona tiene un equipo y la computación ubicua, donde cada persona comparte muchos equipos. La era actual de Internet es vista como una fase de transición entre la del PC y la ubicuidad.

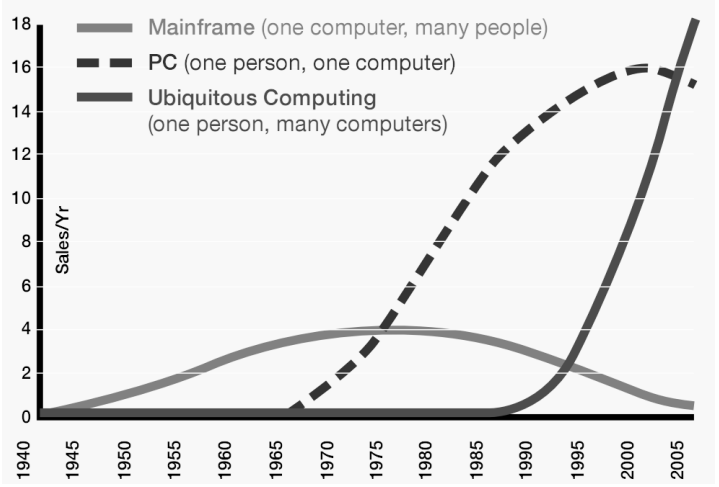


Figura 1. Las tres eras de la Computación. (Weiser, 96)

Al lograr esta visión de transformación, se tendrá profundas implicaciones en la forma como las personas viven, trabajan, interactúan y aprenden. En (Weiser, 1991) la visión sobre el poder de la computación embebida en objetos, lugares y dispositivos, hace un tiempo parecía ser lejana y hasta utópica. Actualmente, con el desarrollo de internet y las tecnologías Web, esto ya no es un sueño. Varios elementos de computación ubicua se están implementando y están colaborando de manera trasparente, natural y útil en diferentes actividades o tareas cotidianas.

Se puede afirmar, que el ánimo de la computación ubicua es copar el ambiente con dispositivos electrónicos, capaces de asistir a los humanos en el desarrollo de sus tareas diarias, de una manera natural y poco intrusiva. En un escenario de estos, los usuarios podrán expresar sus deseos o necesidades y el ambiente se configurará de manera autónoma para satisfacerlos. Las personas interactuarán con una gran cantidad de dispositivos inteligentes de una forma totalmente transparente, sin tener conciencia de la presencia o localización de los dispositivos, y sin manejar interfaces específicas y/o complejas (Bottaro, et al., 2007).

Es importante que la propuesta de una red ubicua haga uso de perfiles de usuario y de capacidades de dispositivo al ofrecer servicios, para que sean más apropiados y puedan ser consumidos por el usuario final. En éste sentido, las características principales de la Computación Ubicua son: i) Transparencia: la computación ubicua debe facilitar la tarea del usuario en una forma no intrusiva, a través de un uso fácil para "ocultar" la tecnología subyacente (Barkhuus, 2002) ii) Movilidad: Los servicios ubicuos deben tener en cuenta el comportamiento de la red de comunicaciones móviles. iii) Contexto de entrega: se define como la propiedad de un sistema que utiliza el contexto para proporcionar información pertinente al usuario, donde la relevancia depende de la tarea o necesidad del usuario (El-Sayed, et al., 2006).

Sin embargo, realizar la visión de un ambiente ubicuo presenta un gran número de problemas debido a la heterogeneidad y dinamismo del ambiente (Ben Mokhtar, 2007). En redes tan dinámicas como las ubicuas, donde no se asegura una disponibilidad permanente de los dispositivos, se tiende a utilizar la computación distribuida aplicada al almacenamiento, procesamiento, etc. (Weiser, 1997) sugiere que los futuros sistemas operativos se basen en micro-núcleos, de modo que puedan configurarse automáticamente para adecuarse de forma dinámica a las necesidades cambiantes del entorno. Éste tipo de dinamismo, requiere la implementación de mecanismos, que faciliten el descubrimiento de nuevas capacidades en la red y su integración de manera autónoma, para brindar una total transparencia al usuario.

Los entornos Ubicuos enfatizan en la necesidad de aplicaciones dinámicas y automáticas, debido a que éste tipo de ambientes se caracterizan por la variabilidad y la movilidad de las entidades que actúan en la red. La entrega de servicios recuperados que se ajusten realmente a las necesidades del usuario es un gran desafío en la computación ubicua. En éste sentido, los mecanismos de descubrimiento de servicios son indispensables en ambientes con estas características, cuyo objetivo es superar la ambigüedad de los servicios recuperados, garantizando su relevancia con relación a la solicitud realizada por el usuario (Bottaro, et al., 2007). Por lo tanto, el descubrimiento de servicios es una etapa muy importante en ambientes de computación ubicua y es de gran utilidad en la generación de nuevos y mejores servicios, que se ajusten a las necesidades de los usuarios.

Considerando un escenario típico de descubrimiento de servicios, el cliente especifica su requisito tomando en cuenta tres aspectos fundamentales (Corrales, et al., 2006): 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 éste vacío).

En la siguiente sección se presentan algunos trabajos con el fin de motivar y contextualizar el presente proyecto de grado.

1.2. Motivación

En esta sección, se expone dos situaciones que requieren de una fase de descubrimiento de servicios en entornos de computación ubicua. La primera está relacionada con e-health y la segunda con entretenimiento móvil.

Aplicaciones e-Health. Para describir este escenario, se ilustra la siguiente situación: Un par de años atrás, Iván fue diagnosticado con Diabetes Tipo II. Con el fin de controlar su nivel de azúcar en la sangre, él lleva un dispositivo de vigilancia que regularmente mide su nivel de azúcar. Ahora supondremos que Iván está en un viaje de negocios en Cartagena. Después de un ajetreado día, de repente se siente mareado. Su terminal e-health identifica una situación anormal e informa a su médico en Bogotá, a varios cientos de kilómetros de distancia. El médico acaba de llegar a casa cuando recibe le mensaje e-health de Iván en su PDA. El médico revisa la información recibida y la historia médica de Iván a través de un servicio web con el fin de tomar decisiones acerca de la condición reportada, posiblemente ordena a través del mismo servicio, algunas pruebas de laboratorio clínico o la realización de una receta médica. Iván recibe la orden de su médico en su teléfono móvil, con las instrucciones correspondientes para la recuperación de la información relacionada con el tratamiento, interactuando con él a través de IPTV y vídeo bajo demanda (Hermida, 2009b).

Aplicaciones de entretenimiento móvil. Los proveedores de redes sociales móviles pueden ofrecer servicios de valor agregado fácilmente si desarrollan y ofrecen nuevos servicios basados en sus propios servicios y en los de terceras partes. En este caso, es necesario que su descubrimiento tenga en cuenta el perfil de usuario y las limitaciones de los dispositivos móviles para mejorar la satisfacción del usuario. Un ejemplo típico puede ser un cliente usando un servicio móvil en una red social, como por ejemplo video-streaming, proporcionado por un tercero y un sistema de votación implementado por su propio proveedor.

1.3. Planteamiento del Problema

El problema abordado en el presente proyecto de grado es: ¿cómo soportar el descubrimiento de servicios en un ambiente de computación ubicua, de manera que los servicios recuperados se ajusten a los requerimientos del usuario?

Precisamente, uno de los objetivos más difíciles de alcanzar en éste tipo ambientes es ayudar a los usuarios en la realización de sus tareas, en cualquier momento y en cualquier lugar, por medio del descubrimiento de los servicios pertinentes ofrecidos en la red ubicua

(Ben Mokhtar, et al., 2006). Por otro lado, la red ubicua debe ser capaz de localizar e invocar los servicios necesarios para satisfacer los requisitos expresados por el usuario. Sin embargo, encontrar un servicio que cumpla exactamente con estas peticiones se convierte en un caso excepcional, la mayoría requieren de una adaptación de los servicios disponibles en la red o un proceso de composición de diversas capacidades para ejecutar tareas complejas. De este modo, en contextos tan dinámicos y heterogéneos como los ambientes ubicuos es imprescindible contar con mecanismos de descubrimiento de servicios que permitan explotar los recursos (aplicaciones, archivos, almacenamiento, hardware, etc.) presentes en la red, ofreciendo la suficiente flexibilidad para reconfigurar los servicios de acuerdo a las variaciones del ambiente. El descubrimiento de servicios deberá incorporar las características necesarias para ser tan dinámico y autónomo como las condiciones de la red ubicua lo requieran.

1.4. Objetivos

El propósito de éste proyecto de grado es adaptar técnicas y algoritmos de emparejamiento para el descubrimiento de servicios en ambientes de computación ubicua, estas técnicas trabajan sobre modelos formales de representación de procesos a nivel atómico y permiten la entrega de coincidencias totales y/o parciales, así como una evaluación de la distancia semántica entre los servicios recuperados y los requeridos por los usuarios. Incluso si no existe un servicio que satisfaga exactamente al usuario, los más similares serán recuperados y propuestos para su posterior uso. El problema de emparejamiento es abordado por medio de una representación formal de grafos, reduciendo de esta forma el problema de emparejamiento de servicios a un emparejamiento de grafos. Lo expuesto anteriormente es ampliado en el siguiente capítulo.

1.5. Contribuciones y Principales Resultados.

Las principales contribuciones de éste proyecto de grado son:

- Este trabajo soporta el componente de descubrimiento de servicios en la tesis de maestría: **Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua** que se desarrolla en la Universidad del Cauca(Hermida, 2009a).

- **Análisis detallado de emparejamiento de servicios.** Los principales resultados de éste análisis son: (a) Análisis de diferentes técnicas usadas para el emparejamiento de servicios, que posteriormente puedan ser adaptadas a ambientes de computación ubicua; (b) Definición y descripción de la representación formal de servicios.

- **Definición de las técnicas y algoritmos de emparejamiento de servicios.** En esta parte una solución para la recuperación de servicios sobre modelos de comportamiento fue implementada, mediante una representación formal de grafos de los servicios. El propósito de esto es utilizar un algoritmo de emparejamiento de grafos a nivel atómico que permita obtener coincidencias totales y/o parciales y así recomendar los servicios más adecuados, que se ajusten a los requerimientos del usuario.

- **Análisis y definición de los repositorios de servicios, usuarios y dispositivos.** Es aquí donde interviene o se introduce el concepto de ubicuidad. Con las diferentes características y dificultades que un ambiente ubicuo presenta, es necesario definir y utilizar repositorios que almacenen 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.

- **Adaptación de la técnica de emparejamiento a ambientes de computación ubicua.** Esta adaptación se logra integrando la técnica de emparejamiento de servicios con los recursos que proveen características ubicuas a los servicios.

- **Prototipo de descubrimiento de servicios en ambientes ubicuos.** Se desarrolló un prototipo llamado U-ServiceMatch, el cual implementa todos los criterios propuestos. Esta herramienta permite la ejecución de los diferentes algoritmos definidos que intervienen en el emparejamiento, y recuperación de servicios en ambientes de computación ubicua.

- **Prototipo para evaluar la eficacia de la técnica de recuperación de servicios.** Se implementó Be4SeD, la cual es una aplicación que permite evaluar la calidad del algoritmo de emparejamiento de servicios en ambientes ubicuos. Esta plataforma de evaluación permite crear un ranking de servicios basada en una comparación manual, por parte de evaluadores expertos en el tema de emparejamiento de servicios, entre un servicio de consulta y los servicios encontrados en el repositorio. La plataforma compara el resultado obtenido por U-ServiceMatch y un ranking definido por los evaluadores.

- Motivar el desarrollo y la investigación sobre los servicios ubicuos en la comunidad académica de la Universidad del Cauca, implementando un prototipo de descubrimientos de servicios ubicuos que cree un precedente alrededor de éste tema, y despertar así el interés por estas tecnologías en la comunidad investigativa del país, debido al positivo panorama social y económico factible de obtener con su difusión.

- En el proceso de desarrollo del presente proyecto de grado, se resalta la publicación de dos artículos: el primero (Hermida, et al., 2010), ya fue evaluado y se encuentra en proceso de publicación en “La Revista Facultad de Ingeniería” de la universidad de Antioquia, clasificada en los índices Latindex y Publindex de COLCIENCIAS en categoría A1. Éste artículo valida la arquitectura de plataforma de descubrimiento de servicios para ambientes ubicuos, soportando los capítulos 3 y 4, relacionados con la adaptación de técnicas de descubrimiento a entornos ubicuos. También evidencia el uso de Be4SeD como herramienta de evaluación de la plataforma propuesta, validando los resultados obtenidos por la técnica de emparejamiento, capítulo 5. El segundo artículo, se encuentra en proceso de evaluación en la revista “Ingeniería y Competitividad” de la Universidad del Valle, clasificada en los índices Latindex y Publindex de COLCIENCIAS en categoría A2, ésta publicación valida la plataforma Be4SeD, soportando el capítulo 5. Estas publicaciones se pueden encontrar en el **Anexo E**.

1.6. Contenido de la Monografía.

Organizada en seis capítulos presentados a continuación:

Capítulo II. Estado del Arte

Presenta una visión general sobre los trabajos relacionados y los conceptos que giran en torno al descubrimiento de servicios en ambientes ubicuos.

Capítulo III. Descubrimiento de servicios en ambientes ubicuos

Primero se realiza un análisis detallado de los modelos formales para representar procesos, enfocando su análisis al descubrimiento de servicios a nivel atómico. Luego, se introduce el problema de emparejamiento de grafos explicando su definición y notación, resaltando las ventajas de utilizar una representación formal para abordar el problema del emparejamiento de servicios. Seguido, se expone el emparejamiento de actividades básicas, obtenidas del proceso de negocio BPEL después de ser transformado a grafo con un ejemplo. La segunda parte de éste capítulo describe como el emparejamiento básico de servicios puede ser adaptado para recuperar servicios en ambientes ubicuos. Para esto, es conveniente utilizar descripciones aplicadas a características de servicios en ambientes ubicuos, en éste caso orientado a redes y dispositivos móviles. Es así como se utiliza los siguientes repositorios: i) Repositorio de servicios: se utiliza el repositorio de procesos de negocio presentado en (Vanhatalo, et al., 2006), el cual soporta el almacenamiento y consulta de documentos BPEL (y otros documentos XML). ii) 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) y WURFL (Wireless Universal Resource) (Passani, 2007). iii) 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. Para finalizar, se presenta un ejemplo de emparejamiento de servicios en un entorno de computación ubicua.

Capítulo IV. Arquitectura Plataforma U-ServiceMatch

En éste capítulo, se expone el proceso de ingeniería realizado para definir la arquitectura que soporta la plataforma de descubrimiento de servicios en ambientes ubicuos, U-ServiceMatch. Obteniendo de las iteraciones planteadas, los Diagramas de Casos de Uso, Diagramas de Clases y Diagramas de Paquetes de las aplicaciones software que constituyen el Sistema.

Entre los aspectos teóricos abordados en esté capitulo se expone las consideraciones previas a la definición de la arquitectura y desarrollo del sistema-solución. Finalmente se presenta el diagrama de despliegue de U-ServiceMatch que describe la configuración de la Plataforma para su ejecución en un entorno real.

Capítulo V. Experimentación y Evaluación

Primero se describe la metodología de evaluación y se presenta la arquitectura genérica, funcionalidades e interfaces de la plataforma Be4SeD desarrollada para evaluar la eficacia

de la técnica de emparejamiento implementada. Be4SeD permite evaluar el emparejamiento de servicios considerando o no el concepto de ubicuidad. Dado que en entornos móviles, los tiempos de respuesta son un factor crítico, este capítulo también presenta la evaluación de eficiencia realizada a la plataforma U-ServiceMatch. Por último, son expuestos los resultados de las evaluaciones de desempeño, describiendo los escenarios y estrategias de prueba consideradas.

Capítulo VI. Conclusiones y trabajos futuros

Presenta las conclusiones y algunas perspectivas de la investigación, analizando los resultados del trabajo realizado y generando un conjunto de recomendaciones importantes para el desarrollo de trabajos futuros.

Capítulo II

ESTADO DEL ARTE

En este capítulo son presentadas las bases teóricas, estándares y especificaciones más relevantes, que permitieron generar una propuesta de descubrimiento de servicios en ambientes de computación ubicua, también se expone el estado actual de diferentes trabajos relacionados con dicha temática. Finalmente se presenta un resumen donde se expone los principales aportes de este capítulo.

2.1. Conceptos Generales

Esta sección contiene conceptos relevantes para la comprensión y posterior análisis del problema central del presente trabajo de grado. Soporta el componente de descubrimiento de servicios en la tesis de maestría: **Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua** (Hermida, 2009a)que se desarrolla en la Universidad del Cauca. Por esta razón, a continuación se expondrá conceptos relacionados con SOA y los entornos Ubicuos, orientados a la composición y especialmente al descubrimiento de servicios.

2.1.1. Computación Ubicua

La visión de Ubicuidad y su concepción fue introducida desde 1988 por Mark Weiser en el CIO del "*Computer Science Laboratory at Xerox PARC*" y fue reconocido mucho después en 1991 mediante su trabajo "*The Computer for the Twenty-First Century*" (Weiser, 1991). Desde entonces, la evolución de la concepción sobre lo que se entiende por ubicuidad parece no haber sufrido cambios. Esta puede ser vista desde dos enfoques completamente diferentes. Por un lado está la visión de la ubicuidad de la tecnología misma: un ejemplo de esto son las comunicaciones satelitales o los servicios como el GPS. Sin embargo esta perspectiva no es la adecuada, la ubicuidad está orientada a la disponibilidad de servicios, procesos e información en todo momento, en cualquier lugar e independientemente del dispositivo de acceso a dicha información *"anytime, anywhere, any device"*.

Para comprender mejor éste concepto, se determinará qué no es ubicuidad. Es decir: La computación ubicua es, por ejemplo, totalmente opuesta a la realidad virtual. Se conoce que la virtualización ubica a las personas en un mundo concebido por un ordenador y traslada sus vidas hasta allí. Por otro lado, la computación ubicua obliga al computador a convivir con las personas en el mundo real. Introduciendo a las TIC en cada aspecto de los procesos culturales, sociales y económicos.

La computación ubicua proporciona una nueva visión en la sociedad, manifestada a través de las mejoras reflejadas en la calidad de vida de las personas. De esta forma se puede concluir que la Ubicuidad es una simbiosis innovadora de servicios, procesos e información que permite, a través de las tecnologías de la información, proveer un escenario sustentable para el desarrollo socio-económico y cultural del mundo.

Es necesario observar la innovación, no desde el punto de vista de la tecnificación, si no como la generación de nuevas capacidades en servicios y procesos. Un ejemplo de ellos podría ser el siguiente: Rastreo o monitoreo por GPS del trasporte público. Sin lugar a duda es algo innovador, ¿Pero dónde está dicha innovación? Si se enfoca en el aspecto técnico, se podría resaltar que está en la incorporación de dispositivos de recepción para GPS en dichas unidades de transporte.

Con relación a lo anterior, los servicios ubicuos se caracterizan por estar basados en condiciones físicas, como el contexto del usuario (localización, tiempo, capacidades de dispositivos, etc), el clima, el estado de ánimo, preferencias de usuarios y flexibilidad. Esta última es una característica muy importante, y alude a que se puede acceder desde cualquier dispositivo, facilitando en gran medida las tareas de las personas.

Ahora bien, siendo técnicos y profundizando, en (Weiser, 1991) se define la computación ubicua como el incremento en el uso de sistemas de cómputo, los cuales pueden estar embebidos en el entorno haciéndolos disponibles e invisibles para los usuarios. Esta propuesta se ha posicionado como la tercera generación o paradigma en la computación y cuenta ya con múltiples aplicaciones y equipos de investigación que procuran su desarrollo (Gimeno, 2004). Se pueden destacar las siguientes características de los ambientes ubicuos (Kenichi, 2004)(Almenárez, 2005).

1. **Invisibilidad**. Weiser propone dispositivos “invisibles” que interactúan con los usuarios sin que estos tengan que preocuparse de cómo hacerlo, más aún, dicha interacción ocurre por debajo del nivel de conciencia de las personas.
2. **Dinamismo**. La libertad de movimiento de los usuarios y el uso de tecnologías inalámbricas hacen que estos ambientes sean altamente dinámicos al permitir que nuevos dispositivos se agreguen o abandonen a la red.
3. **Espacios colaborativos**. La arquitectura para entornos ubicuos está basada en dispositivos autónomos que interactúan en el espacio físico colaborativamente para dar soporte a los usuarios presentes en la red.
4. **Escalabilidad local**. Los usuarios disponen de capacidades según el contexto en el que se encuentren, por lo tanto, el concepto de servicios locales es muy importante en computación ubicua, ya que algunos servicios pierden todo sentido cuando el usuario se encuentra fuera de un entorno determinado.
5. **Enlace transparente entre redes**. De acuerdo con la red disponible, la distribución de los servicios ofrecidos puede variar, por lo tanto, estos servicios deben ofrecerse de forma continua frente las distintas redes por donde transite el usuario. Existe entonces una alta heterogeneidad de las tecnologías integradas en términos de la red, dispositivos e infraestructuras software.

2.1.2. SOA (Service Oriented Architecture)

Es un concepto arquitectural para el desarrollo de sistemas altamente autónomos y con un bajo acoplamiento que son capaces de comunicarse, componerse y evolucionar en ambientes abiertos, dinámicos y heterogéneos tales como los ambientes ubicuos(Georgantas, y otros, 2005).

El estilo arquitectónico de SOA está basado en tres roles principales: el Proveedor del Servicio, asumido por una entidad que ofrece un servicio; Cliente del Servicio, entidad que desea consumir el servicio ofrecido; y el Registro de Servicio, rol asumido por la entidad que mantiene la información acerca de los servicios disponibles y la forma de accederlos. En (Papazoglou, 2003)se introduce un nuevo rol dentro de SOA denominado Agregador de Servicio, su función es la de componer nuevos servicios a partir de los existentes y ofrecerlos a las aplicaciones clientes. El Agregador actúa como un Proveedor, ofreciendo los servicios compuestos, y como un Cliente al consumir los servicios existentes. En la Figura 2 se ilustran las relaciones existentes entre los roles de SOA.

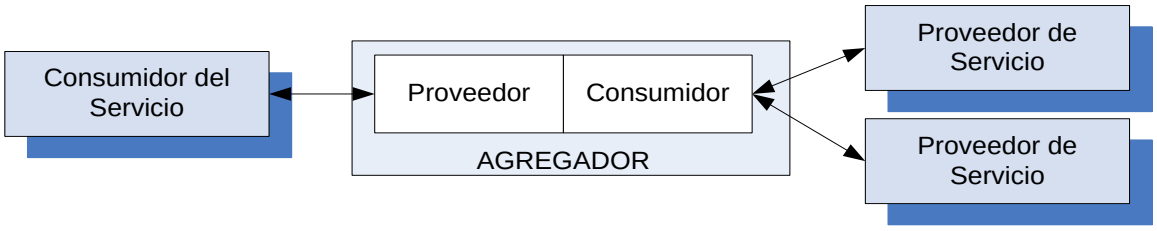


Figura 2. Elementos Conceptuales de SOA

En la Figura 2 se presenta en detalle las funciones desempeñadas por cada rol de SOA y las relaciones entre ellos. Los servicios se exponen utilizando una descripción formal que permite su publicación por parte de los proveedores y su posterior localización y consumo por parte de los clientes. En (Ben Mokhtar, 2007)se describen los elementos conceptuales de SOA de la siguiente manera:

- Publicación del Servicio: permite a los proveedores registrar sus servicios en un registro de servicios.
- Localización de Servicio: permite a los clientes encontrar los servicios deseados en el registro de servicios.
- Matching[1] de Servicios: funcionalidad ofrecida por el registro de servicios, permite seleccionar los servicios que mejor respondan a las peticiones realizadas por los clientes.
- Acceso a los Servicios: permite a los clientes establecer una conexión con los servicios seleccionados.
- Composición de Servicios: permite la integración de múltiples servicios para conformar un servicio compuesto.

Una vez identificados los elementos conceptuales asociados al paradigma de SOA se puede realizar una abstracción de estos componentes para soportar el diseño de sistemas ubicuos Figura. Los dispositivos ubicuos son representados como Proveedores de Servicios y corresponden a las entidades que proveen las capacidades necesarias para realizar las tareas de usuario. Los dispositivos ubicuos realizan los anuncios de funcionalidad para alertar a los demás dispositivos de las capacidades que ofrecen, estos avisos corresponden a una Publicación de Servicio. Por otra parte los usuarios del ambiente ubicuo corresponden a un Cliente SOA, y son ellos quienes deben localizar las funcionalidades ofrecidas en el ambiente ubicuo para realizar sus tareas. Las tareas de usuario a su vez son abstraídas

[1] Matching: es el proceso de comparar las peticiones de servicio con las descripciones publicadas y determinar qué servicio satisface de mejor manera la solicitud recibida (Bandara, et al., 2007).

como un Servicio Compuesto, que integra las funcionalidades ofrecidas por los dispositivos presentes en la red.

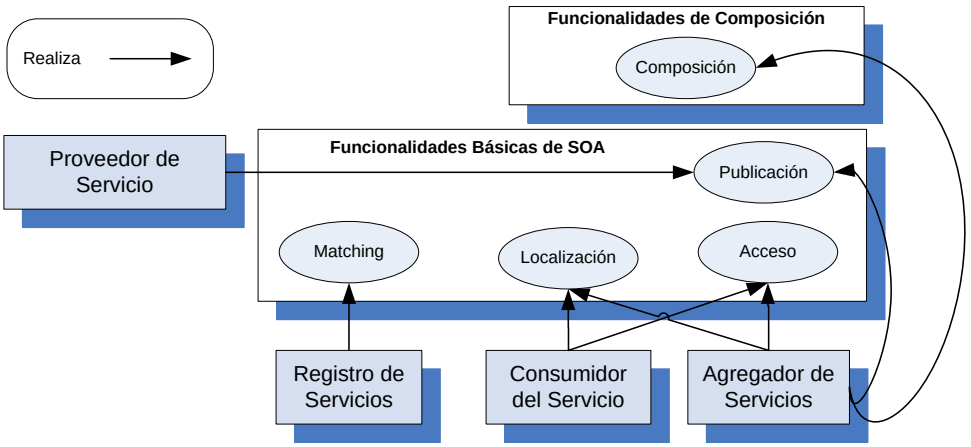


Figura 3. Elementos Conceptuales de SOA

Para la creación autónoma de Servicios Compuestos en las redes ubicuas las funcionalidades de composición deben poseer las siguientes características(Ben Mokhtar, y otros, 2005):

1) Ser dinámica, de acuerdo a los componentes disponibles en un lugar y tiempo específicos;
2) Satisfacer los requisitos funcionales de una manera eficiente dentro de los límites impuestos por las restricciones de recursos de los dispositivos móviles;
3) Acomodarse a la heterogeneidad de las tecnologías

Estas cualidades exigen que los ambientes ubicuos posean las capacidades necesarias para descubrir e integrar los servicios disponibles en la red, independientemente de la movilidad y heterogeneidad de los dispositivos.

2.1.3. Composición de Servicios en Ambientes de Computación Ubicua

En la tesis de maestría: **Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua**(Hermida, 2009a), que se desarrolla en la Universidad del Cauca, se aborda el estudio de una arquitectura de referencia para la composición autónoma de servicios en ambientes de computación ubicua, teniendo en cuenta las características descritas al final de la sección anterior. En éste apartado se expondrá el papel que juega nuestro presente proyecto de grado dentro de la misma.

En (Hermida, 2009b) se describe la arquitectura propuesta en la tesis de maestría, considerando una fase de descubrimiento de servicios previa a la composición. El objetivo de la plataforma es proveer un servicio automático de composición en Ambientes Ubicuos. El usuario aloja en su dispositivo una aplicación conocida como *Solicitante (Requester)*. El *Solicitante* utiliza un lenguaje común de descripción de procesos de negocio para representar los servicios requeridos por el usuario. Esta descripción creada por el *Solicitante* (*Query*) es enviada a la plataforma. Esta recibe la solicitud del usuario y transforma a una representación formal de procesos de negocio para realizar la fase de descubrimiento.

Adicionalmente, El contexto del *Solicitante* es obtenido para encontrar los servicios más pertinentes. Luego la plataforma compone el servicio requerido por el usuario utilizando los servicios descubiertos. Finalmente, el proceso de negocio compuesto por la plataforma es enviado al *Solicitante*.

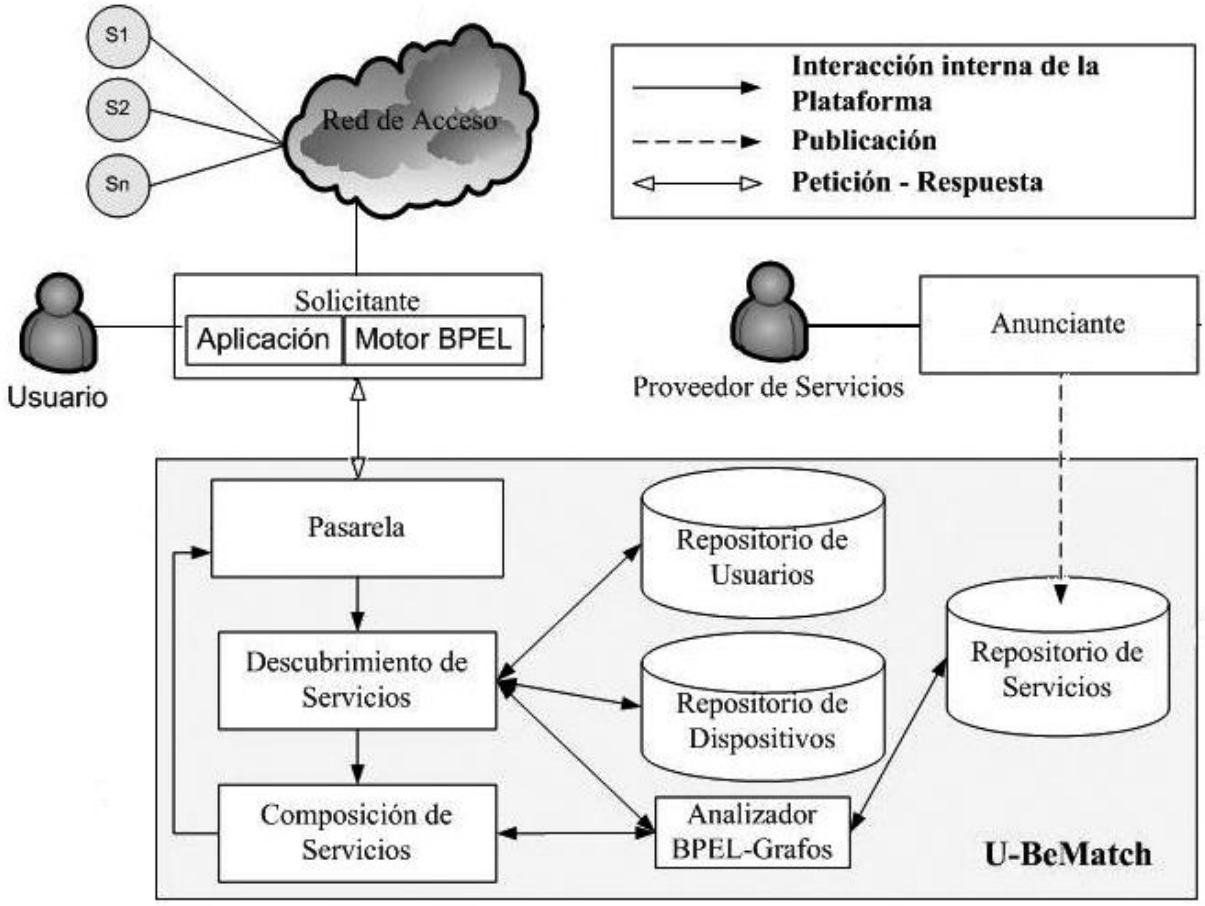


Figura 4. Plataforma U-BeMatch

En la Figura 4 se expone los módulos que conforman la plataforma de composición, estos son: *Requester, Advertiser, Gateway, Service Composition, Parser, Service Registry, Device Repository, User Repository* y *Service Discovery*. En la siguiente sección nos enfocaremos en describir el módulo de Descubrimiento de Servicios, que es el centro de nuestro trabajo de investigación.

2.1.4. Descubrimiento de servicios

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 (Bandara, et al., 2007). El descubrimiento de servicios puede ser abordado principalmente bajo dos enfoques: descubrimiento sintáctico y semántico.

El descubrimiento sintáctico se basa en técnicas de comparación de interfaces (por ejemplo: WSDL[2], IDL[3], interfaces RMI[4], etc.) o palabras clave para realizar la búsqueda de servicios, requiriendo coincidencias exactas a nivel sintáctico entre las descripciones de los servicios y los parámetros empleados (Ben Mokhtar, 2007)(Corrales, y otros, 2008).

La descripción semántica de servicios, busca la posibilidad de que las máquinas puedan realizar un mayor procesamiento y razonamiento sobre los servicios. La principal diferencia con las descripciones sintácticas es la forma como se representa la información. Mientras la sintáctica se enfoca en definir los servicios a partir de los mensajes de entrada y salida, tipos y partes de los mensajes, la semántica busca ofrecer información acerca de la funcionalidad del servicio (Ben Mokhtar, 2007)(Corrales, y otros, 2008). La representación semántica del contenido de las descripciones de un servicio permiten a las máquinas entender y procesar su contenido, soportando el descubrimiento e integración dinámica de los servicios (Ben Mokhtar, et al., 2006). Además el descubrimiento de servicios puede ser clasificado como Centralizado vs. Descentralizado y Sintáctico vs. Semántico.

Se observa actualmente que los enfoques descentralizados y semánticos son los más adecuados para entornos web y su extensión a ambientes ubicuos, pero la aplicación de razonadores semánticos en estos entornos aumenta la cantidad de procesamiento requerido para una respuesta, incrementando el tiempo de ejecución, siendo éste uno de los factores críticos en éste tipo de entornos. También en (Köning-Rics, 2007) se expresa que el desarrollo de tecnologías basadas en semántica, ha sido raramente aplicado debido principalmente a dos razones: Primero, un prerrequisito para usar tecnologías semánticas es la existencia de ontologías comprensibles de dominio. Para muchas aplicaciones, estas ontologías aún no existen y el costo de usar una tecnología semántica, crear una ontología de dominio, puede apreciarse como extremadamente alto. La segunda, viene como una consecuencia del despliegue de tecnologías basadas en Inteligencia Artificial y su asociación a fallas pasadas, por eso las tecnologías semánticas son relegadas con un cierto grado de escepticismo por muchos usuarios potenciales.

2.1.5. Servicios Web

Servicios Web son aquellos componentes por medio de los cuales diferentes sistemas exponen sus aplicaciones, permitiendo la creación de nuevos Servicios Compuestos. Su importancia se destaca en entornos distribuidos, por eso a continuación definiremos algunas de sus características para entender su papel en la creación de nuevas funcionalidades.

Inicialmente tenemos que los servicios Web son definidos como aplicaciones software modulares e independientes, las cuales pueden ser descritas, publicadas, localizadas e invocadas a través de una red, estas buscan la interoperabilidad entre distintas aplicaciones (W3C, 2010). La arquitectura de un servicio Web está compuesta por una serie de protocolos. Entre los más importantes esta SOAP, el cual se basa en XML para el intercambio de información en un ambiente distribuido, WSDL que define un esquema XML para la descripción de la interfaz, semántica y administración de llamada a un servicio web y UDDI define como publicar y descubrir información acerca de los servicios Web.

[2]Web Services Description Language, es un formato XML que se utiliza para describir servicios Web.

[3]Interface Description Language, es un lenguaje de especificación de interfaces que se utiliza en software de computación distribuida.

[4]Remote Method Invocation.

En el marco de SOA también encontramos la composición de servicios Web, que permite la implementación de procesos de negocio interoperables. Este apartado es de gran importancia, ya que el presente trabajo de grado da soporte al componente de descubrimiento de servicios en la tesis de maestría: **Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua**(Hermida, 2009a)**.** La composición de servicios Web implica hallar mecanismos que permitan a dos o más servicios cooperar entre sí, para resolver requerimientos que van más allá del alcance de sus capacidades individuales (Álvarez, et al., 2005), dos términos son comúnmente usados al referirse a composición o colaboración de servicios, estos son *Orquestación de servicios* y *Coreografía de servicios*.

Se considera orquestación de servicios, cuando los servicios Web son controlados por una sola entidad, en éste tipo de proceso la entidad que está orquestando es la única que conoce la información y el flujo del proceso orquestado. En sí, un solo proceso utiliza y manipula la información de los diferentes servicios, por ejemplo la salida de uno, la puede convertir en la entrada de otro.

La coreografía de servicios se da cuando se define colaboración entre diferentes aplicaciones, sin importar la plataforma en que se esté ejecutando, o el lenguaje en que éste definida. A diferencia de la orquestación, en la coreografía una entidad no controla a todos los participantes, aquí se define un comportamiento común que todos los participantes deben conocer.

2.1.6. Procesos de negocio

La Coalición para la Gestión de Workflow (Workflow Management Coalition - WfMC) (WfMC, 2008), define a un proceso de negocio como “un conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos”, también se define a un proceso de negocio como un servicio Web complejo formado por la relación lógica de funciones suministradas por diferentes servicios Web, los cuales ya existen en la Web y están dinámicamente integrados para garantizar una tarea de negocio más compleja (Mongiello, et al., 2006).

La utilidad los procesos de negocio se observa especialmente en la etapa de solicitud de servicios por parte del usuario (Hermida, 2009b). En ella, el usuario describe que acciones desea realizar, luego el *Requester* traduce sus requerimientos a un documento que contiene la descripción de un proceso de negocio. Lo anterior, le permite al usuario definir el comportamiento, la secuencia, el tipo de servicios a consumir. El usuario también puede enviar solicitudes de procesos de negocio creados anteriormente, sin importar el dispositivo. Posteriormente, el módulo de descubrimiento de servicios se encarga de encontrar los componentes pertinentes adaptando la búsqueda al dispositivo que en ese momento utilice para tener acceso a la red.

2.1.7. Estándares para la ejecución de procesos

Para la ejecución de procesos de negocio se han definido varios estándares y lenguajes entre los cuales se destacan *Electronic Business XML (ebXML)* (ebXML, 2006), *Yet Another Workflow Language* (YAWL) (YAWL, 2010)(W3C, 2010),Web Services Choreography Description Language(WS – CDL) (W3C, 2004), Business Process Modeling

Language(BPML), Web Services Flow Language (WSFL)(Cover, 2002)y Business Process Modeling Language(BPEL) (Andrews, et al., 2003) .

De estos estándares el presente proyecto de grado seleccionó BPEL puesto que contiene características útiles en la composición de servicios web, como: el manejo de contextos (actividad Scope), la manipulación de fallas, manejo del comportamiento del sistema y el modelamiento de colaboración entre socios, que brindan varias funcionalidades al momento de su aplicación en entornos distribuidos, especialmente entornos de computación ubicua. Además, el hecho de que BPEL sea un estándar de facto en el campo de composición de servicios, brinda la ventaja de tener un modelo teórico que se aproxima más a la industria.

2.1.7.1. Lenguaje para la ejecución de proceso de negocio (Business Process Execution Language – BPEL)

BPEL es un lenguaje para la ejecución de procesos de negocio (Andrews, et al., 2003), que permite definir una notación estándar, para especificar el comportamiento de un proceso basado en servicios. En sus inicios era considerado un lenguaje que solamente manejaba el secuenciamiento de un solo proceso, donde los procesos de negocios de las empresas eran, menos complejos que los que hoy en día operan, debido a que antes las organizaciones solo interactuaban dentro de su intranet. Posteriormente, las organizaciones comenzaron a definir actividades más complejas en las cuales más de un participante estaba incluido en el proceso de negocio, estos participantes exponían al entorno web lo que podían aportar haciendo uso de servicios web. Es por ello que BPEL migra hacia su versión BPEL para servicios web o como su sigla lo indica, BPEL4WS, el cual es usado para el presente proyecto.

BPEL4WS consiste en un lenguaje de orquestación basado en XML, diseñado para el control centralizado de la invocación de diferentes servicios Web, que define el orden preciso en el cual un servicio Web debe ser invocado, generando que los procesos descritos en BPEL4WS exporten e importen funcionalidades usando solo interfaces de Servicios Web.

BPEL4WS permite especificar *procesos abstractos* y *procesos ejecutables*. Los procesos abstractos especifican la secuencia de mensajes que deben ser intercambiados por varias entidades que participan en un proceso de negocio, pero sin revelar aspectos internos de implementación en cada entidad. Es decir los procesos abstractos especifican coordinaciones de servicios Web. Un proceso ejecutable, por el contrario, especifica el comportamiento interno de una entidad, de tal forma que pueda ser ejecutado automáticamente por un motor de ejecución, por lo tanto los procesos ejecutables especifican composiciones de servicios Web.

En la Figura se presenta varios componentes definidos en BPEL, cada uno tiene su propia funcionalidad.

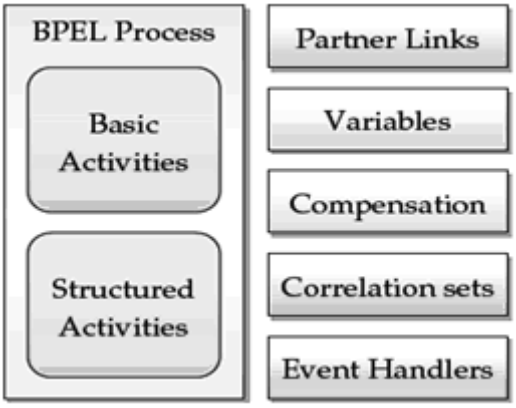


Figura 5. Elementos principales de un proceso BPEL

- **Actividades:** realizan la lógica del proceso, estas representan el tipo de interacciones que se llevan a cabo internamente en la empresa, las cuales se dividen en dos tipos, actividades básicas y estructuradas.

    - **Actividades Básicas:** describen pasos elementales del comportamiento del proceso. Estas son:

        - **<receive>:** Espera por la llegada de un mensaje externo de petición para una operación de un servicio Web proporcionado por el propio proceso.
        - **<invoke>:** Invocación de una operación de un servicio Web proporcionado por una entidad externa al proceso.
        - **<assing>:** Asigna un valor a una o más variables.
        - **<wait>:** Permite a un proceso de negocio especificar un tiempo de espera, antes de realizar una acción, por ejemplo, hasta la invocación de un servicio web específico.
        - **<terminate>:** Termina la ejecución del proceso de negocio.
        - **<empty>:** Permite a un proceso de negocio especificar un tiempo durante el cual no se haga nada.

    - **Actividades Estructuradas:** contienen internamente a otras actividades y simplemente especifican restricciones que aplican a dichas actividades. Estas actividades, permiten estructurar el proceso BPEL ayudando a mantener el flujo del proceso. Algunas actividades estructuradas son:

        - **<sequence>:** Contiene una colección de actividades a ser ejecutadas secuencialmente.
        - **<flow>:** Especifica un conjunto de actividades que deben ser ejecutadas concurrentemente.
        - **<pick>:** La actividad pick espera que ocurra uno de varios eventos y ejecuta la actividad asociada a dicho evento.
        - **<while>:** Ejecuta iterativamente una o varias actividades, mientras se cumpla una condición.
        - **<switch>:** Permite la ejecución de actividades siempre y cuando se cumplan ciertas condiciones.

- **PartnerLinks**: BPEL permite describir relación entre los socios del proceso (procesos que interactúan con otros). Los servicios Web siempre son modelados como partnerLink, cada proceso BPEL tiene al menos un partner, el cual generalmente tiene que ser un cliente que invoca el proceso. La etiqueta <partnerLinks> es usada para describir los servicios socios.

- **Variables:** empleadas durante la ejecución del proceso BPEL. Estas se asocian con tipos de mensajes que pueden ser especificados como entradas o salidas para las actividades receive, invoke y reply, por medio de las variables se mantienen los mensajes intercambiados entre socios del negocio, como también mensajes usados solamente dentro del proceso.

- **Compensation:** proceso por el cual se invierte o se provee una alternativa para restaurar los datos que una actividad hubiera realizado. Por medio de **Compensation** se da marcha atrás a procesos de negocio completados o actividades ejecutadas, cuando ocurren problemas.

- **Correlation Sets:** permite la asociación de un mensaje entrante con una instancia del proceso, esto es posible gracias a la existencia de identificadores únicos para cada solicitud o mensaje recibido. También permite especificar grupos de operaciones correlacionadas en una instancia de un servicio web.

- **Event Handlers:** cuando un evento específico ocurre, el proceso cambia al código que maneja el evento. Hay dos clases de eventos a los que un proceso BPEL puede reaccionar:

    - Eventos de Mensaje (*Message events*) <*onEvent*>, puede ser comparado a la etiqueta <receive> en la cual se está a la espera de un mensaje entrante.
    - Eventos Alarm (*Alarm events*), <*onAlarm*>, es usada cuando se necesita esperar cierta cantidad de tiempo antes de que algo ocurra.

  Los manejadores de eventos se diferencian de las actividades normales en que ellos son ejecutados solamente cuando ocurre un evento. Si no ocurre un evento el proceso lo abandona y continúa la ejecución.

- **Partner Link Types**

Caracterizan la relación conversacional que pueda existir entre dos servicios, definiendo los roles que juega cada uno en la conversación. Cada rol especifica exactamente un WSDL portType. La etiqueta <partnerLinkType> representa el rol de cada socio del servicio en la comunicación.

- **Port Types**

Es un concepto ligado al estándar WSDL, que muestra las operaciones soportadas y ofrecidas por un Servicio Web que referencia el documento WSDL. En la definición de un portType, figura el nombre del Servicio Web seguido de las funciones que éste puede ofrecer.

En el resto del presente documento se escribe BPEL para referirse a la versión BPEL4WS que es la utilizada para el presente trabajo de grado.

2.2. Trabajos relacionados

Existen diferentes trabajos que abordan los núcleos temáticos del presente proyecto, en ellos se han presentado propuestas interesantes, alrededor de diferentes aspectos involucrados en el descubrimiento de servicios en ambientes ubicuos. A continuación se presenta un análisis de los mismos, exponiendo su aporte y aplicación al momento de desarrollar este proyecto de grado.

2.2.1. Descubrimiento de Servicios en ambientes Ubicuos

De la sección 2.1.5 se tiene que el descubrimiento de servicios está definido como: la capacidad de encontrar y utilizar posteriormente un servicio, basado en alguna descripción publicada de su funcionalidad y parámetros operacionales (Bandara, et al., 2007). A continuación serán expuestos enfoques desde los cuales éste concepto ha sido aplicado, como etapa previa a la composición de servicios en entornos de computación ubicua.

- En (Sellami, et al., 2009) se aborda el descubrimiento de servicios en ambientes ubicuos orientado al uso de la semántica y el contexto, éste trabajo propone registrar los servicios de forma distribuida en redes P2P. Estos registros son estructurados de acuerdo a la capacidad y dominio semántico de los servicios publicados. Para describir las capacidades, (Sellami, et al., 2009) propone extender la WSDL añadiendo más atributos a las descripciones, la descripción obtenida se denomina CSWSDL, esto permite fortalecer la capacidad de WSDL. En el trabajo de (Sellami, et al., 2009) la idea de usar P2P para contrarrestar problemas de la arquitectura cliente servidor, relacionados con escalabilidad, se ve comprometida por el incremento en la complejidad para encontrar recursos; además, en la actualidad la gran mayoría de dispositivos no cuentan con la suficiente capacidad para soportar tales procesos.

- En(Corrales, 2008) se desarrolla una técnica de emparejamiento de procesos de negocio, permitiendo obtener coincidencias parciales y una evaluación de la distancia semántica entre los servicios encontrados y los requisitos de los usuarios. Este trabajo aborda el problema de emparejamiento de servicios por medio de una representación formal de grafos y propone un algoritmo para lograr el matching aproximado de servicios, tomando al emparejamiento de servicios como una etapa necesaria en el proceso de descubrimiento de servicios. Esta propuesta ha sido probada en el contexto de los Servicios Web por medio del prototipo Ws-BeM (*Web services-Behavioral Matchmaking).*

- Al interior del Grupo de Ingeniería Telemática de la Universidad del Cauca, se han adelantado otros proyectos alrededor de los ambientes ubicuos. La tesis de maestría “Marco de Referencia para la Construcción de Aplicaciones de Comercio Electrónico Móvil en Países en Vía de Desarrollo” (Martinez, 2008) define un conjunto de lineamientos articulados para la construcción de aplicaciones móviles, los cuales se enfocan en la adaptación de contenido teniendo en cuenta las características de los dispositivos terminales; de este modo sirve de base para la implementación de un ambiente de computación ubicua, que contemple aspectos adicionales como los perfiles y la localización de los usuarios. El trabajo de grado “Protocolo de Descubrimiento e

Interacción de Servicios Ubicuos en un Ambiente Móvil" (Fajardo, et al., 2008) aborda el descubrimiento de servicios ubicuos en ambientes móviles, lo cual fue de gran ayuda en el presente trabajo.

- La tesis de (Ben Mokhtar, 2007) introduce un modelo semántico, orientado a los servicios de middleware para la computación ubicua, soportando la especificación de dos tipos de descripciones de servicios, semántica y sintáctica. Esto permite a proveedores de servicios y solicitantes proporcionar especificaciones semánticas más exactas, lo cual facilita que el middleware propuesto realice un matching de servicios más preciso. (Ben Mokhtar, 2007) presenta un prototipo denominado mediador PERSE (*Pervasive Semantic-aware Middleware*) el cual incluye un módulo para el descubrimiento de servicios en ambientes ubicuos, empleando modelos basados en conversaciones. Los algoritmos empleados en éste mediador describen las conversaciones como un autómata de estados finitos, lo cual limita la expresividad de los modelos, siendo su principal problema la ejecución en paralelo de diferentes capacidades ya que genera modelos demasiado extensos.

- En (Steller, et al., 2009) (Steller, et al., 2009) y (Steller, et al., 2006) se desarrolla una técnica de matching de servicios reemplazando solicitudes basadas en significados sintácticos por solicitudes basadas en significados semánticos, incluyendo información del contexto, mejorando considerablemente el resultado de la búsqueda de los servicios solicitados. En éste trabajo se aprecia la utilidad de un modelo que utilice el "contexto" para enriquecer el proceso de matching, durante el descubrimiento de servicios en ambientes ubicuos, haciendo uso de OWL-S para publicar y solicitar los servicios almacenados en un repositorio de descripción de servicios, además de éste repositorio, se utilizan dos repositorios más, de usuarios y de capacidades de dispositivos. Se encuentra también que el modelo propuesto, para introducir características de contexto, todavía no ha sido validado por un prototipo, quedando sus resultados en teoría. Finalmente vale la pena resaltar que Steller descubre interfaces WSDL más no actividades dentro de un proceso de negocio.

- En (Doulkeridis, et al., 2006) se propone una arquitectura centralizada de descubrimiento de servicios. Esta utiliza un registro de servicios considerando el contexto. Para descubrir un servicio, dos consultas son utilizadas *Qusr* y *Qcxt*. *Qusr* Extrae los servicios usando palabras clave especificadas por un consumidor de servicios. Estos servicios son transmitidos al registro de servicios sensible al contexto. *Qcxt* se utiliza para elegir los servicios emparejados considerando el contexto del consumidor.

- En (Arabshian, et al., 2006), los autores proponen una arquitectura distribuida basada en GloServ (Arabshian, et al., 2004) que clasifica los servicios de acuerdo a su dominio de negocio y asocia un agente de contexto para cada clase. Cuando un agente recibe una consulta, éste busca las clases más apropiadas para ello, y reenvía la consulta a los agentes que den esas clases. Uno de los agentes le preguntará al usuario por su contexto y busca el servicio más adecuado

- El trabajo presentado en (Sivashanmugam, et al., 2004) se basa en el uso de registros federados. Una federación es un conjunto de registros cooperantes y autónomos que actúa sobre un dominio específico de negocio. Mediante la estructuración de la red de registros en las federaciones, se reduce el número de registros consultados en un

proceso de descubrimiento. Las anotaciones semánticas para WSDL son utilizadas para describir servicios semánticos.

- En (Xu, et al., 2007), semánticamente describe las funcionalidades de un servicio usando OWL-S (Martin, et al., 2004). Los servicios están distribuidos de acuerdo a la descripción textual contenida en su etiqueta WSDL <documentation>. En el descubrimiento de servicios, una consulta es enviada primero al grupo más similar, luego, una correspondencia entre las necesidades del usuario y los servicios encontrados en ese grupo es realizada. Para lograr esto, la consulta es enviada a todos los servicios del grupo.

- En (Wombacher, et al., 2004) los autores brindan una semántica formal al emparejamiento de procesos de negocio basados en autómatas de estados finitos, agregando expresiones lógicas a cada uno de los estados. Sin embargo la complejidad computacional es muy alta y no es muy escalable para repositorios con muchos servicios. En (Wombacher, et al., 2005) se agrega un índice basado en sistemas de base de datos tradicionales que reduce la complejidad computacional. Por otro lado, la representación de autómata de estados finitos limita la expresividad de los modelos, siendo su principal problema la ejecución en paralelo de diferentes capacidades ya que genera modelos demasiado extensos. El trabajo de (Shen, et al., 2005) emplea una representación de autómatas finitos no-deterministicos para modelar el comportamiento de los servicios web, emplea OWL-S para describir el perfil de las actividades de cada servicio, logrando capturar las propiedades semánticas y temporales del comportamiento de los servicios. Estas soluciones no ofrecen un resultado positivo en casos donde no se encuentra un modelo que corresponda exactamente al requerido por el usuario. Además, asumen que los servicios operan con una semántica común para describir los mensajes intercambiados, lo cual limita la efectividad al momento de comparar la similitud entre dos servicios.

- En (Hermida, 2009b) se propone una plataforma para la composición de servicios en ambientes ubicuos basado en comparaciones de servicios y contexto. Propone al igual que (Corrales, 2008) un matching basado en modelos de comportamiento. En su propuesta se resalta la aparición de módulos dedicados al perfil de usuario y a la descripción del dispositivo utilizado para acceder a la plataforma de composición.

2.2.1.1. Análisis

La propuesta presentada en (Corrales, 2008) puede ser adaptada a las condiciones de ambientes ubicuos y ser una alternativa viable para el desarrollo de la técnica de emparejamiento dentro de éste proyecto. Sin embargo, para descubrir servicios, (Corrales, 2008)realiza un emparejamiento a nivel de procesos de negocio, haciendo más complejo el proceso de emparejamiento de servicios, comprometiendo el tiempo de ejecución al aumentar el procesamiento requerido. Por ésta razón, el presente proyecto de grado plantea un descubrimiento basado en un emparejamiento atómico de servicios.

En otros trabajos tales como el mediador PERSE (*Pervasive Semantic-aware Middleware* presentado por (Ben Mokhtar, 2007) se puede observar que los algoritmos empleados para el desarrollo de un mediador describen las conversaciones como un autómata de estados finitos, lo cual limita la expresividad de los modelos, siendo su principal problema la ejecución en paralelo de diferentes capacidades ya que genera modelos demasiado

extensos. En este proyecto(Ben Mokhtar, 2007) se utiliza Autómatas de Estados Finitos, lo mismo que en (Wombacher, et al., 2004) y (Shen, et al., 2005), entre otros; observando que diversos enfoques orientados al descubrimiento de servicios buscan representar sus requerimientos y recursos por medio de modelos formales de representación. Igualmente, en (Corrales, 2008) se resalta el uso de un modelo formal para representar procesos BPEL, en este caso Grafos. Así, en la sección 3.1.1se expondrá un estudio sobre modelos formales de representación para la posterior elección de uno de ellos en esta tesis.

En (Sellami, et al., 2009) se utiliza WSDL para describir los servicios, la cual es extendida a CSWSDL para describir las capacidades de los dispositivos, logrando que el descubrimiento de servicios sea eficiente, al considerar el contexto. (Steller, et al., 2006) Al igual que el anterior trabajo, se enfoca principalmente en el descubrimiento de servicios, teniendo en cuenta la semántica y el contexto, para enriquecer el proceso de descubrimiento en un entorno ubicuo, haciendo uso de OWL-S para publicar y solicitar los servicios almacenados en un repositorio de descripción de servicios. También (Doulkeridis, et al., 2006) propone una arquitectura sensible al contexto para soportar el descubrimiento, considerando los requisitos del usuario *Qusr* y refinando esa búsqueda utilizando el contexto del consumidor *Qctx*. En (Arabshian, et al., 2006), la arquitectura propuesta en el presente trabajo de grado, contiene un agente encargado de buscar el servició más adecuado dependiendo del contexto del usuario.

En estos trabajos se muestra una tendencia a personalizar la búsqueda de servicios, como un paso hacia la adaptación del descubrimiento a los entornos de computación ubicua. Esta personalización tiene en cuenta dos tipos de información, que en conjunto describen el contexto del solicitante: las preferencias de usuario y el contexto de entrega del servicio requerido. La inclusión del contexto en el proceso de descubrimiento induce un cambio en la representación de los servicios, al adicionar una descripción de las características necesarias para ser consumidos.

Así, los aspectos relacionados con la personalización del descubrimiento de servicios aplicada a un entorno de computación ubicua serán abordados en la sección2.2.2.

2.2.2. Personalización

El paradigma de Personalización se enfoca en adaptar aplicaciones tanto como sea posible a las preferencias de usuario y al contexto del usuario. La personalización de datos, hace referencia al conjunto de técnicas que permiten proveer al usuario los datos más relevantes, dependiendo de su criterio (Abbar, et al., 2008). La necesidad de tener en cuenta la movilidad y omnipresencia (Rudi Belotti, 2004) ha impuesto nuevas consideraciones tales como la localización de usuario, el medio utilizado para la interacción y muchas otras características agrupadas en el concepto de Contexto de Usuario. Un contexto es un conjunto de características, las cuales describen el entorno en el cual el usuario interactúa con el sistema de información. Mientras un perfil, es el conjunto de especificaciones que caracterizan las necesidades del usuario en términos de datos. A continuación se abordará la personalización del descubrimiento de servicios basados en esas dos consideraciones: Perfil y Contexto de Usuario

Perfil de usuario: en (Abbar, et al., 2008) se han definido cinco dimensiones a través de las cuales un perfil de usuario puede ser definido: Datos personales, Dominios de Interés, Calidad de los datos, datos entregados y Seguridad y privacidad. La dimensión del dominio de interés (DoI) describe la experiencia o calificación tan bien como el contenido en el cual

está interesado. La dimensión de datos personales reúne información acerca del usuario mismo (Identidad, datos demográficos, etc). La dimensión de Calidad de datos agrupa Factores de calidad que definen la calidad esperada por los usuarios. La dimensión de Seguridad y Privacidad, describe las reglas de seguridad y restricciones que pueden ser aplicadas ya sea a los resultados de datos de las consultas, a las consultas mismas, a la identidad del usuario o a todo el perfil como una información sensible. Finalmente, la dimensión de entrega de datos contiene características de los datos entregados e información acerca de la interacción del usuario (e.g. Formato, número de resultados entregados, display layout).

Contexto de usuario: desde que las preferencias de usuario pudieron cambiar con relación a su interacción con el entorno (lugar, momento, disponibilidad, etc.), se ha definido un contexto, como el conjunto de información que caracteriza esos entornos de usuario. Se han identificado cinco dimensiones para el meta modelo del contexto: temporal, espacial, equipo, estado de usuario y entorno. La dimensión temporal agrupa atributos relacionados al aspecto temporal de la interacción del usuario. Esta información permite personalizar una aplicación con respecto al momento de la interacción con el usuario. La dimensión espacial describe la localización geoespacial del usuario durante la interacción. También indica si el usuario esta estático o en movimiento. La dimensión de equipo caracteriza al medio por el cual el usuario interactúa con la aplicación; encierra información acerca del dispositivo del usuario, acerca del software del usuario y acerca de la conexión. La dimensión del estado del usuario provee información acerca de la disponibilidad del usuario y emociones cognoscitivas. Finalmente, la dimensión de entorno concierne a características y medidas de varios sensores tales como los de temperatura o luminosidad (Abbar, et al., 2008).

Al buscar información sobre la aplicación de los anteriores conceptos en trabajos relacionados con el descubrimiento de servicios, se encuentra que el trabajo de (Steller, et al., 2006) define un Repositorio de Usuarios y otro de dispositivos. Estos repositorios también son presentados en la arquitectura propuesta en (Hermida, 2009b). Finalmente los dos trabajos exponen en sus arquitecturas la existencia de un repositorio de servicios que almacena las descripciones de los servicios publicados y sus especificaciones adicionales. A continuación se estudiará los repositorios de usuario, dispositivos y servicios, analizando su aplicación en el presente proyecto, como herramientas para la personalización del descubrimiento de servicios teniendo en cuenta el contexto y el perfil del usuario.

2.2.2.1. Repositorio de Usuarios

Almacena detalles relacionados al usuario del dispositivo solicitante. Estos pueden incluir, preferencias explicitas, provistas por el usuario cuando él o ella utilizaron el sistema por primera vez o datos implícitos recolectados por la plataforma. Los datos implícitos, pueden incluir patrones de uso histórico y comportamiento del usuario tales como los servicios previamente invocados por el usuario (Steller, et al., 2006). De esta manera en éste repositorio se abarca las características relacionadas con el perfil de usuario, como fue descrito en la sección 2.2.4.1, también alberga información sobre las dimensiones temporal, espacial, estado de usuario y entorno del usuario.

2.2.2.2. Repositorio de dispositivos

Cada usuario estará asociado con uno o más dispositivos. Cada dispositivo tiene sus propias especificaciones, tal como capacidad de procesador, tamaño de pantalla, interfaces, etc. Esta información será tomada del dispositivo, cuando él se registre con el sistema por

primera vez (Steller, et al., 2006). Éste repositorio almacena especificaciones del dispositivo, tales como capacidad de procesamiento, modalidades de presentación y sus parámetros tanto en entradas como en salidas. Esta información puede ser recolectada de i) cabeceras-HTTP, estas cabeceras permiten detectar el tipo de contenido y el agente de usuario utilizado durante el acceso del dispositivo a un servicio basado en http. ii) UAProf (User Agent Profile). Se utiliza para capturar clases de capacidades de dispositivo y preferencias de información. Las cuales incluyen la plataforma de hardware y software del dispositivo, también información sobre la red a la cual el dispositivo está conectado y iii) WURLF (Wireless Universal Resource) es un archivo de configuración XML, el cual contiene información acerca de las capacidades y características de diversos dispositivos móviles. Este proyecto es de fuente abierta y es propuesto por desarrolladores que trabajan con WAP y Wireless.

Al analizar las definiciones del repositorio de dispositivos, su aplicación está relacionada con brindar la información de los dispositivos utilizados para el acceso a la red. A partir de las especificaciones brindadas por éste repositorio, se puede refinar la búsqueda para garantizar que los servicios recuperados puedan ser consumidos. (Steller, et al., 2006) Propone el almacenamiento de las relaciones entre dispositivos y usuarios en éste repositorio. Por otra parte (Hermida, 2009b) lo utiliza como una base de información para identificar al dispositivo utilizado. En el presente trabajo de grado se tomó la decisión de dejar a éste repositorio las funciones que define (Hermida, 2009b), y adoptar las sugerencias hechas por (Steller, et al., 2006). Pero en lugar de almacenar estas relaciones en el repositorio de servicios, se hará como una extensión del repositorio de usuario. Así éste repositorio es el encargado de suministrar la información relacionada con la *dimensión de dispositivo* contenida en el contexto del usuario.

2.2.2.3. Repositorio de servicios

Este repositorio almacena las descripciones de los servicios (Target Services) y metadatos asociados. Los datos en el repositorio son organizados de un modo jerárquico; esto permite consultar el repositorio implementando el primer paso en el proceso de descubrimiento. Los servicios son filtrados, para así reducir el número de candidatos para el segundo paso en el proceso de descubrimiento: el emparejamiento (Hermida, 2009b). En éste repositorio serán especificados los atributos dinámicos del servicios (Steller, et al., 2006).

De acuerdo a lo anterior, éste repositorio almacenará básicamente las descripciones de los servicios y sus características adicionales. Es importante aclarar que estas características son las **requeridas** para la ejecución del servicio, mientras las características almacenadas en los repositorios de usuario y dispositivos, son aquellas que el usuario posee para consumir el servicio. Así, para garantizar que un servicio pueda ser consumido las segundas deben superar las primeras, o al menos cumplirlas.

Para el presente proyecto es indispensable usar un repositorio de servicios flexible, que permita almacenar documentos BPEL y otro tipo de archivos basados en el estándar XML, que enriquecen el proceso de negocio con características de contexto (*requerimientos de contexto del servicio,* entregados por los proveedores o desarrolladores de los servicios). A continuación son presentados diferentes sistemas de almacenamiento

- Sistema de Archivos (p.ej. en Windows XP)
- Sistema de Control de Versiones (p.ej. CVS)
- Bases de Datos XML (p.ej. Natix (Fiebig, et al., 2002))

- Bases de Datos relacionales (p.ej. DB2)

Los tipos de almacenamiento de datos presentados anteriormente son construidos para usos genéricos, soportando el almacenamiento de diversos tipos de datos. Los sistemas de archivos, sistemas de control de versiones y bases de datos XML están construidos para mantener los datos organizados en archivos. Las Bases de datos XML proveen servicios adicionales para el manejo de documentos XML, a diferencia de los sistemas de archivos normales y los sistemas de control de versiones, que no tienen servicios específicos para manipular éste tipo de registros. Ninguna de estas soluciones soporta documentos BPEL o WSDL. Las bases de datos relacionales son construidas para un almacenamiento de datos eficiente (buen desempeño en las consultas y recuperación de información) superando a los demás. Sin embargo, se requiere de un tipo diferente de organización de datos, usando una representación relacional en duplas. Los datos se organizan en tablas, que pueden tener índices para hacer las consultas más eficientes. Por lo tanto, no tienen un soporte nativo de archivos XML.

Soporte de Archivos XML:

Conceptualmente hay dos métodos para almacenar datos XML: no estructurados y estructurado(McCown, 2004). El enfoque no estructurado guarda un documento XML completo como una sola unidad, el cual es utilizado por los sistemas de archivo y de control de versiones. Las Bases de datos relacionales soportan éste enfoque, al permitir que el documento XML sea almacenado como un único tipo de datos CLOB (*Character Large Object*). La consulta no estructurada de documentos XML es ineficiente, porque todo el documento debe ser transformado para la consulta. La estructura del documento XML no puede ser utilizado para cargar sólo las partes del documento pertinentes a la consulta.

Algunas bases de datos relacionales han comenzado recientemente a ofrecer un almacenamiento XML estructurado, también conocido como soporte XML nativo. El enfoque estructurado permite consultas eficientes y mantiene la jerarquía de los datos. Las bases de datos relacionales comerciales, como IBM DB2 8.1, Oracle 10g, Sybase y ASE 12.5.1, soportan el almacenamiento estructurado de archivos XML. Sin embargo, en el momento, solamente Oracle soporta la consulta eficiente de contenidos con XQuery para XML. Las bases de datos XML, como Natix (Fiebig, et al., 2002) de la Universidad de Mannheim, proporcionan almacenamiento XML estructurado. Al igual que Oracle, Natix soporta XQuery. Desde que Oracle 10g es un producto comercial, Natix es más adecuado para prototipos de investigación académicos por ser *open-source*.

Repositorio de Procesos de Negocio BPEL:

En (Vanhatalo, et al., 2006) se presenta un repositorio que permite almacenar procesos de negocio y metadatos asociados. Este proyecto es un *plug-in* de Eclipse construido originalmente para procesos de negocio BPEL y otros datos XML relacionados. Además, provee un framework que facilita el almacenamiento, búsqueda y uso de éste tipo de documentos. El repositorio puede ser fácilmente extendido con nuevos tipos de archivos XML. Este proyecto proporciona un API de Java que facilita manipular el código XML como objetos, siendo trasparente para el usuario la serialización y de-serialización del documento. Permitiendo que el usuario pueda manejar los datos más convenientemente con objetos; estos datos almacenados cumplen con el estándar del esquema XML. La información puede ser consultada como objetos Java utilizando OCL (lenguaje de consulta

orientado a objetos). El diseño flexible de esta implementación permite que el motor de consulta OCL sea sustituido por otro basado en un lenguaje de consulta diferente.

Análisis:

Los autores de (Vanhatalo, et al., 2006) exponen que no hay repositorios BPEL públicamente disponibles, los cuales puedan ser comparados con el enfoque presentado por ellos. Resaltan que solo existe almacenamiento de archivo XML genéricos, donde los documentos BPEL pueden ser almacenados, sin embargo su posterior uso es de un alto costo. Argumentan que las aplicaciones que utilizan actualmente archivos BPEL como objetos, no tienen un repositorio de procesos BPEL, como una capa separada, sino que cada aplicación se encarga de la persistencia de los datos de forma individual.

Además, ninguna de las formas de almacenamiento de datos presentadas anteriormente, provee enlaces relacionados directamente con la localización de los archivos almacenados, estos vínculos pueden ser implementados sobre ellos (otro nivel) por una determinada aplicación.

En el contexto de los procesos de negocio, es relevante tener un eficiente mecanismo de consulta. Los sistemas de archivos y sistemas de control de versiones no los proveen. Por ello, es importante considerar funcionalidades de búsqueda de archivos que puedan ser usadas en el almacenamiento y manipulación de información en éste tipo de sistemas. Esto facilitaría la búsqueda de un archivo que contenga un determinado *substring*. Sin embargo, la simple búsqueda de un *substring* es propensa a errores, ya que el usuario se interesa más en la búsqueda de la cadena en un contexto específico y no en cualquier parte del archivo. Por lo tanto, éste tipo de búsqueda, puede dar lugar a interpretaciones erróneas de los datos.

El Lenguaje de Ejecución de Procesos de Negocios para Servicios Web, es el estándar de facto de la industria para representar procesos de negocios. El cual está directamente relacionado con otros estándares, como el Lenguaje de Definición de Servicios Web y XML. Adicionalmente, metadatos arbitrarios representados en formato XML pueden ser asociados a los procesos de negocio dependiendo del contexto y la aplicación donde sean usados. Lo anterior es una importante característica considerada por el presente proyecto de investigación. Por lo tanto se utilizará el repositorio BPEL expuesto en (Vanhatalo, et al., 2006) como el repositorio de servicios.

2.3. Resumen

Este capítulo presentó el estado actual sobre investigaciones desarrolladas alrededor del descubrimiento de servicios en ambientes de computación ubicua, comenzando por contextualizar al lector, brindándole una descripción de conceptos generales que posteriormente le permitieron comprender el campo de aplicación del presente proyecto.

Seguido, en la sección de trabajos relacionados se argumentó por qué para el proceso de recuperación o descubrimiento de servicios para ambientes de computación ubicua, es de gran importancia una fase adecuada de emparejamiento de servicios que trabaje a nivel atómico sobre las abstracciones formales de los procesos de negocio, ya que no es conveniente trabajar soportando un emparejamiento complejo de procesos de negocio, dado que el costo de procesamiento y tiempos de ejecución pueden ser elevados, lo cual no es

aplicable en este tipo de entornos donde los bajos tiempos de respuesta son un factor crítico que debe cumplirse.

Por último, se abordó el problema de la personalización del descubrimiento de servicios, llegando a la justificación de la utilización de tres tipos de repositorios: Repositorio de usuarios, Repositorio de dispositivos y Repositorio de Servicios. Así, en el desarrollo de éste documento se extenderá más su análisis y se expondrá su aplicación en entornos de computación ubicua.

Capítulo III

DESCUBRIMIENTO DE SERVICIOS EN AMBIENTES UBICUOS

En el estado del arte se mostró que la adaptación del descubrimiento de servicios a ambientes de computación ubicua, implica la adopción de dos conceptos fundamentales: el emparejamiento de servicios y la personalización. Para el emparejamiento, se destacó la propuesta de varias investigaciones que abordan este problema desde el tratamiento de modelos formales de representación de procesos, esta discusión es profundizada en la sección 3.1. Por otro lado, la sección 3.2 aborda la aplicación de la personalización en el proceso de descubrimiento, analizándolo bajo los conceptos de perfil y contexto de usuario. Antes de abordar los temas mencionados, se describirá de manera general nuestra propuesta para el proceso de descubrimiento de servicios en ambientes ubicuos, Figura 6.

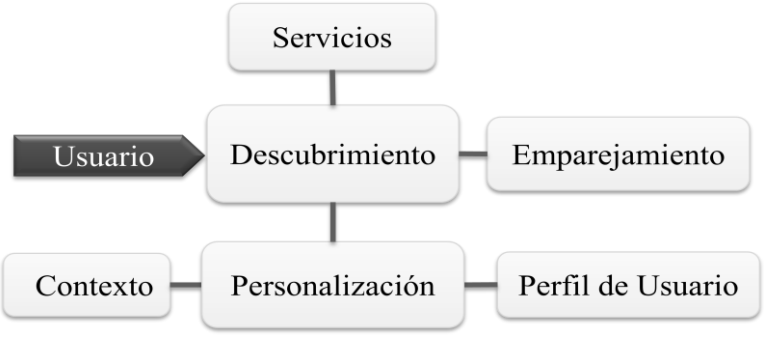


Figura 6. Modelo conceptual básico

El usuario solicita un servicio con los requisitos funcionales que desea. Al entrar al sistema se identifica las características de contexto, enriqueciendo con ellas la descripción del servicio requerido por el usuario. Luego basado en la descripción del perfil de usuario, el sistema realiza una fase de recomendación de servicios candidatos, para determinar su similitud utilizando al emparejamiento. Si los servicios recomendados no cumplieran con características aceptables en el valor de su similitud, se realizará el emparejamiento directamente con los servicios del repositorio de servicios. Por último, a partir de las características de contexto, se presenta al usuario un listado organizado de los servicios pertinentes y los servicios sugeridos. Estos últimos, a diferencia de los primeros, son servicios recomendados que satisfacen la necesidad del usuario, pero no pueden ser consumidos desde su dispositivo actual. Esta sugerencia es realizada ya que un usuario puede tener uno o más dispositivos para acceder al sistema.

3.1. Emparejamiento

En el estado del arte del presente proyecto de grado se presentaron diversos trabajos, relacionados con el descubrimiento de servicios en ambientes ubicuos, que proponen el uso

de modelos formales de representación de procesos. Así, esta sección presenta la justificación de la elección de grafos como modelo formal de representación de procesos de negocio BPEL. Posteriormente se expone la técnica seguida en la transformación de BPEL a grafos. Una vez claras las equivalencias entre estas dos representaciones, se presenta una propuesta de emparejamiento de servicios basada en el trabajo descrito en (Corrales, 2008). Propuesta que define un conjunto de algoritmos, que operan sobre una representación formal de grafos en el proceso de emparejamiento de actividades básicas. Finalmente un ejemplo del emparejamiento de actividades básicas BPEL es presentado.

3.1.1. Modelos Formales para Representación de Procesos

La importancia de utilizar una representación formal de procesos en el descubrimiento, enfocado a la composición de servicios en ambientes ubicuos, es considerada en múltiples investigaciones. Así, considerando el estudio realizado en(Corrales, 2008), a continuación se presenta la evaluación de cuatro modelos formales de representación de procesos: Redes de Petri, Autómatas de estados finitos, modelos de Algebra de Procesos y Grafos. Cada uno de ellos ofrece diferentes ventajas al momento de describir el comportamiento de procesos.

3.1.1.1. Evaluación de los Modelos Formales

En esta sección se presentará la evaluación de los modelos formales mencionados. El objetivo de esta comparación es proporcionar criterios para modelar procesos a través de una notación formal, que facilite su posterior uso y se adapte adecuadamente a las demandas del presente trabajo de grado. Aspectos importantes a considerar para ello son: la integridad (un completo conjunto de semántica y madurez en la notación), compatibilidad (una propiedad que permite razonar acerca de un sistema compuesto sobre la base de sus componentes sin necesidad de información adicional de la implementación de sus partes), paralelismo (es un aspecto clave de formalismo que debe cumplirse para modelar con precisión los procesos) y la complejidad para implementar el matching (matching polinomial).

La Tabla1 resume las representaciones formales de los procesos explicados anteriormente y que pueden ser profundizados en (Corrales, 2008)(Milner, 1982)(Peterson, 1981)(Hopcroft, y otros, 2001)(Bunke, 2000). El símbolo (+) significa que el parámetro en la columna es soportado por el modelo formal. El símbolo (-) significa que la propiedad no es soportada. El signo (+/-) significa que la propiedad es moderadamente soportada. Por último, el símbolo N/A es no aplicable, es decir, la propiedad no se aplica al modelo con el cual está siendo asociada.

Representación Formal	Parámetros de Evaluación			
	Integridad	Compatibilidad	Paralelismo	Matching Polinomial
Modelos de Algebra de Procesos	+	+	+	N/A
Redes de Petri	+	-	+	+
Autómatas de Estados Finitos	+	+	-	+
Representación Grafos	+	+	+	+/-

Tabla 1. Evaluación de Modelos Formales de Representación de Procesos

Por lo tanto, se concluye y considerado lo expuesto en(Corrales, 2008), que el modelo de representación de grafos ofrece una simple y madura notación para expresar la semántica de un servicio. La representación de grafos permite analizar un sistema compuesto a nivel de sus partes, sin la necesidad de adicionar información acerca de la implementación de ellas.

Además, el paralelismo soportado por el modelo de grafos es un aspecto clave de formalismo que debe cumplirse para modelar con precisión el emparejamiento de servicios y por ende su posterior uso tanto en el descubrimiento de servicios, como en la composición de los servicios recuperados. El emparejamiento de grafos, a pesar de ser un proceso complejo, brinda técnicas que se pueden adaptar con el fin de aumentar y mejorar su rendimiento, ver(Christmas, y otros, 1995), (R.Wilson, y otros, 1994), (Branca, y otros, 2000). Este tipo de representación, es utilizado en varios enfoques como una representación formal de proceso de negocio (Mendling, y otros, 2005), (Pautasso, y otros, 2003), (Gerbe, y otros, 1998).

3.1.1.2. Representación de Grafos

Los grafos son una general y poderosa estructura de datos para la representación de objetos y conceptos. En la representación de un grafo, los nodos suelen constituir objetos o partes de objetos, mientras que las aristas describen las relaciones entre los objetos o partes de objetos. Los Grafos tienen algunas propiedades interesantes de invariancia. Por ejemplo, si un gráfico es dibujado en un papel, es trasladado, rotado o transformado en su imagen espejo, en sentido matemático éste sigue siendo el mismo grafo. Éstas propiedades de invariancia, así como el hecho de que los grafos se adaptan perfectamente a modelos de objetos en términos de partes y de sus relaciones, los hace muy atractivo para diversas aplicaciones (Bunke, 2000). En aplicaciones tales como el reconocimiento de patrones y computación visual, la similitud entre objetos es una característica relevante. Dada una base de datos de objetos conocidos y una consulta, la tarea es recuperar uno o varios de los objetos de la base de datos que sean similares a la consulta. Si los grafos se utilizan para representar objetos, éste problema se convierte en determinar la similitud de grafos, que generalmente se conoce como matching de grafos.

3.1.2. Transformación de BPEL a Grafos

En este apartado se describirá la equivalencia entre una descripción BPEL y una representación formal de Grafos para procesos de negocio, encontrada en(Corrales, 2008). Un grafo tiene un nodo de comienzo y puede tener múltiples nodos de finalización. Un grafo también posee dos clases de nodos: los nodos regulares, que representan las actividades y los conectores, que representan las reglas de separación o juntura del tipo XOR o AND. Los nodos son conectados mediante aristas, las cuales pueden tener un *guard* opcional. Los *guards* son condiciones que pueden ser evaluadas como verdaderas o falsas.

En esta transformación, se implementa la estrategia de barrido presentada en (Mendling, y otros, 2005) para transformar un documento BPEL a un grafo BPEL. La idea general del barrido es mapear Actividades Estructuradas (*SA*) a sus respectivos Fragmentos de Grafo BPEL (ver Algoritmo 1). Éste algoritmo recorre la estructura anidada del flujo de control BPEL (BCF) de arriba hacia abajo y aplica recursivamente un proceso de transformación específico para cada tipo de actividad estructurada. Primero, los nodos *Start* y *End* son identificados en el documento BPEL, luego por cada actividad estructurada la *Structured Activity graph* (*SAgraph*$_i$) es calculada ejecutando la función *BCF transform* en *SA*$_i$ (mirar algoritmo 2). Seguido el *SAgraph*$_i$ es adicionado al *BPEL graph*. Por cada *SAgraph*$_i$ adicionado al *BPEL graph,* el algoritmo 1 calcula sus aristas con otras actividades estructuradas de grafos (*SAgraph*$_j$) considerando el mapeo *tc*, el cual es definido de acuerdo con las condiciones de transición de las Actividades Estructuradas (*SA*) en el documento BPEL.

Algoritmo 1. Function FlatteningBCF

INPUT: BPELdocument
OUTPUT: BPELgraph

ADD Start node to BPELgraph
ADD End nodes to BPELgraph
for each SA_i **do**
    Calculate $SAgraph_i$ = BCF transform(SA_i)
    ADD $SAgraph_i$ to BPELgraph
    Calculate the edge ($SAgraph_i$,SAgraphj) = tc(SA_i , SAj)
    ADD the edge ($SAgraph_i$,$SAgraph_j$) to BPELgraph
end for
returnBPELgraph

El Algoritmo 2 presenta el procedimiento *BCFtransform*, el cual es invocado recursivamente en elementos anidados. En este algoritmo, el primer parámetro representa la actividad estructurada a ser procesada, seguida por el nodo *predecessor* y *successor* del grafo estructurado de actividades (*SAgraph*), entre los cuales, la estructura anidada está anclada (i.e. predecesor y sucesor).

El procedimiento *BCFtransform,* empieza comprobando si la actividad estructurada actual (SA) sirve como objeto y fuente de relaciones. Si es así, respectivamente serán adheridos conectores al comienzo y al final del bloque actual de la actividad. De este modo las relaciones en el control de flujo de BPEL son mapeadas a aristas y las respectivas conexiones de *join* o *split,* son adicionadas alrededor de las actividades básicas anidadas. Existen cinco sub-procedimientos para manejar las cinco actividades estructuradas: *Secuence, Flow, Switch, While, y Pick*. En éste punto, se asume que Pick es únicamente utilizada para modelar eventos alternativos de comienzo. La transformación de *Scopes* simplemente llama al procedimiento para esa actividad anidada. Actividades *Empty* son mapeadas a una arista entre el nodo predecessor y el nodo successor. *Terminate* es mapeada a un evento *End*. Además, cada actividad Básica BPEL (*Receive, Reply, Invoque , Wait*) es transformada a un nodo de grafo BPEL. Puesto que el control de flujo BPEL es de importancia, la actividad *Assign* es mapeada a una arista entre nodos *predecessor* y *successor*.

Algoritmo 2. Function BCFtransform

INPUTS: *SA*
OUTPUT: *SAgraph*

Calculate *BCF transform*(*SA, Predecessor, Successor, Connector*)
if∃(*$Link_i$, SA*) **then**
    ADD a *$SplitConnector_i$*
    ADD an edge between the *P redecessor Activity* and the *$SplitConnector_i$*
end if
if∃(*SA, $Link_j$*) **then**
    ADD a $JoinConnector_j$ ($Link_j$)
    ADD an edge between the *$JoinConnector_j$* and *Successor activity*

end if
if*activity*∈*Seq***then**
Calculate BCFtransformSeq(SA,Predecessor,Successor)
else if *activity* ∈ *F* low **then**
Calculate BCFtransformFlow(SA,Predecessor,Successor,AND)
else if *activity* ∈ *P ick***then**
Calculate BCFtransformPick(SA,Predecessor,Successor,XOR)
else if*activity* ∈ *Switch***then**
Calculate BCFtransformSwitch(SA,Predecessor,Successor,XOR)
else if *activity*∈*W hile***then**
Calculate BCFtransformWhile(SA,Predecessor,Successor,XOR)
else if *activity* ∈ *Basic***then**
Calculate (Activity,Predecessor,Successor)
else if *activity* ∈ *Empty***then**
Calculate (Predecessor,Successor)
else if*activity* ∈ *Assign***then**
Calculate (Predecessor,Successor)
else if *activity* ∈ *Terminate***then**
Calculate (Predecessor,End)
end if
return*SAgraph*

Los procedimientos de *BCFtransformSeq*, *BCFtransformFlow*, *BCFtransformPick*, *BCFtransformSwitch* y *BCFtransformWhile,* generan los elementos del grafo BPEL que corresponden a sus respectivas actividades estructuradas BCF. *BCFtransformSeq* transforma una *Sequence,* conectando todas las actividades anidadas con las aristas del grafo. Aunque no esté definida explícitamente, esta transformación requiere un orden definido en las actividades anidadas. Para cada sub-actividad el procedimiento *BCFtransform* es invocado nuevamente. La representación de grafos de *Switch* (*BCFtransformSwitchprocedure*), en el grafo BPEL, consiste de un bloque de ramas alternativas entre una separación XOR y una juntura XOR. Las condiciones de ramificación son asociadas a las aristas.

La actividad *Pick* tiene algunas similitudes a la *Switch*. Pero, en lugar de evaluar una expresión, espera por la ocurrencia de una fuera de un conjunto de eventos y ejecuta las actividades asociadas. Esos eventos pueden ser relacionados a tiempo o a mensajes recibidos. Sintácticamente, el *BCFtransformPick procedure* mapea a los mismos elementos de control de flujo correspondientes a la actividad *Switch*. En el caso de las condiciones de *OnMessage,* el mensaje es especificado con elementos de flujo no controlable, similares a una actividad *Receive*. En el caso de un evento *OnAlarm,* el tiempo es modelado de la misma manera que para una actividad *Wait*. Cada evento alternativo es seguido por actividades anidadas combinadas con una juntura XOR. El *BCF procedure* es transformado a un bloque de ramas paralelas, comenzando con una separación AND y sincronizada con una juntura AND. Para la actividad *While*, *BCFtransformWhile* crea un bucle entre una juntura XOR y una separación XOR. La condición es adicionada a la arista. Para la transformación de Scopes simplemente se llama al procedimiento de esta actividad anidada.

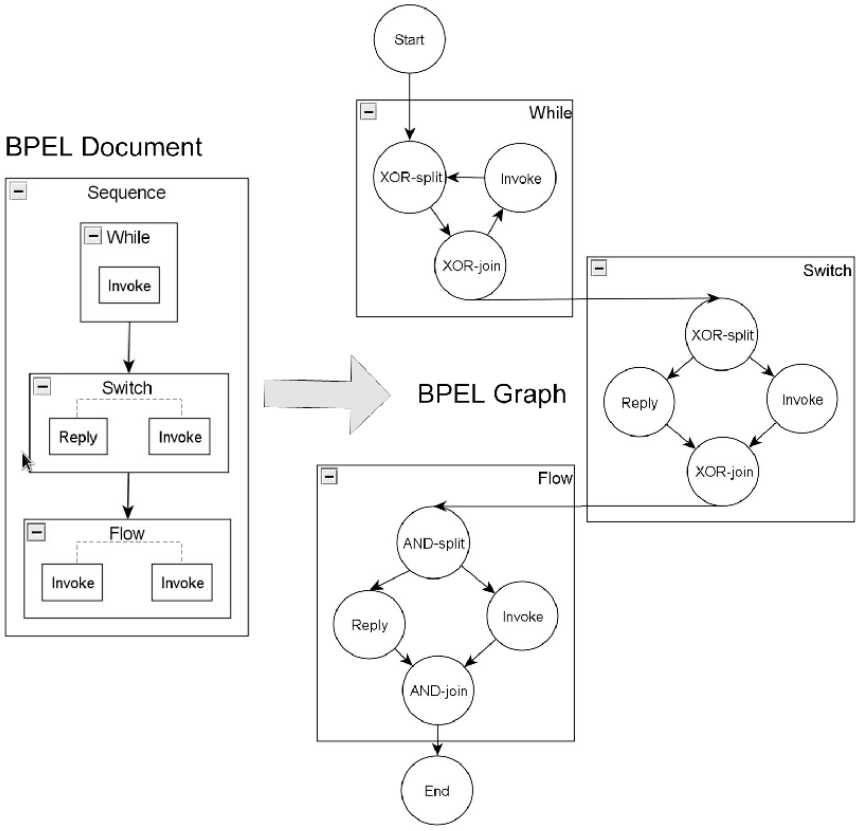


Figura 7. Correspondencia entre elementos BPEL y elementos de Grafos

Los nodos que representan las actividad básicas tienen los siguientes atributos: *ActivityType*(*AT*)*, Operation* (*Op*), *PortType* (*PT*) y*PartnerLink* (*PL*). Los nodos conectores tienen dos atributos: ConnectorType (AND-split, AND-join, XOR-split, XOR-join) y ActivityType (la actividad estructurada BPEL desde la cual fue transformada). La Figura 7muestra la correspondencia entre un esquema BPEL y elementos de grafos.

3.1.3. Emparejamiento de Actividades Básicas

En esta sección, se evidencia la aplicación de un modelo formal de representación de procesos en el emparejamiento atómico de actividades básicas. Como referente teórico se tomó el trabajo realizado en (Corrales, 2008), donde se define un conjunto de algoritmos que permiten calcular la similitud entre dos grafos. Así, a continuación se expondrá los algoritmos que soportan éste proceso, presentando la modificación realizada, para que el algoritmo de emparejamiento trabaje sobre los nodos que representan las actividades básicas, y no con el resto de elementos que componen al proceso BPEL.

Antes de ejecutar un algoritmo de emparejamiento de los nodos de los grafos (G_Q y G_T), es necesaria una etapa previa encargada de la clasificación, organización y filtrado de estos, de acuerdo a su tipo de actividad. De esta forma, sólo los nodos que pertenecen al mismo tipo de actividad en G_Q (*grafo Query*) y G_T (*grafo Target*), respectivamente, son comparados, (es decir: el conjunto de actividades $Invoke_{syn}$, $Invoke_{asyn}$, *Receive* y *Reply*). El algoritmo propuesto, inicia comparando el primer nodo incluido en el primer conjunto de G_Q con todos

los nodos del mismo conjunto de G_T, procediendo de la misma forma para los demás nodos (Algoritmo 3).

Algoritmo 3. Clasificar nodos por tipo de actividad – ClassifierNodes

INPUTS: BPELGraph: Struct (*Star, Receive, Invoke$_{syn}$, Invoke$_{asyn}$, Reply, End*)
OUTPUT: set of nodes organized by Activity Type: *setInvoke$_{syn}$, setInvoke$_{asyn}$, setReceive and setReply*
BEGIN
FOR each activity node *n* in *BPELGraph* **DO**
  if $n \in Star$ **then**
    not consider
  else if $n \in Receive$ **AND** n $\notin setReceive$ **then**
    add *n* to *setReceive*
  else if $n \in Invoke_{syn}$ **then AND** $n \notin setInvoke_{syn}$ **then**
    add *n* to setInvoke$_{syn}$
  else if $n \in Invoke_{asyn}$ **then AND** n $\notin set\ Invoke_{asyn}$ **then**
    add *n* to setInvoke$_{asyn}$
  else if $n \in Reply$ **then AND** n $\notin setReply$ **then**
    add *n* to *setReply*
  else if $n \in End$ **then**
    not consider
  end if
END FOR

Algoritmo 4. Emparejamiento de Actividades Básicas – BasicActivityMatch

INPUTS: (Node$_i$, Node$_j$) Node$_i$: Struct (Op_i, PT_i, PL_i, AT_i), Node$_j$: Struct (Op_j, PT_j, PL_j, AT_j)
OUTPUT: DistanceNode
BEGIN
Calculate Operation Similarity *SimOperation = LS(Op_i, Op_j)*
if SimOperation = 0 (different Operations) **then**
  return DistanceNode = 1
else
  Calculate PortType Similarity *SimPortType = LS(PT_i, PT_j)*
  Calculate PartnerLink Similarity *SimPartnerLink = LS(PL_i, PL_j)*
  CalculateDistanceNode

$$\text{DistanceNode} = 1 - \frac{w_{op} * \text{SimOperation} + w_{pt} * \text{SimPortType} + w_{pl} * \text{SimPartnerLink}}{w_{op} + w_{pt} + w_{pl}}$$

end if

Una vez obtenidos y organizados los nodos, tanto de G_Q como de G_T, La función que permite compararlos es definida mediante el Algoritmo 4, tomado de (Corrales, 2008) y modificado

para que también considere el parámetro *PartnerLink.* Este algoritmo toma como entradas dos nodos, que representan actividades básicas de BPEL (*Receive, Invoke, Reply*), y calcula la distancia semántica entre ellos. Cada nodo posee cuatro atributos: *ActivityType* (*AT*), *Operation* (*Op*), *PortType* (*PT*) y *PartnerLink* (*PL*), de los cuales tres son considerados en el proceso de emparejamiento (*Op, PT y PL*),ya que el parámetro *ActivityType* permite identificar que los nodos a comparar sean del mismo tipo.

La función *BasicActivityMatch*, inicia dando prioridad a la comparación de *Op*, si las dos operaciones son similares (SimOperation > 0), se continua con cálculo de la similitud de los demás parámetros (*PT* y *PL*) para estimar la distancia entre las dos actividades, *DistanceNode*. Los pesos *Wop*, *Wpt y Wpl* indican la contribución de la similitud de los atributos *Op, PT*y *PL* a la similitud de las actividades (0 ≤ *Wop* ≤ *1, 0* ≤ Wpt ≤ *1* y *0* ≤ Wpl ≤ *1*). Para calcular la similitud de los atributos se emplea la función *Linguistic Similarity* (*LS*), descrita en el Algoritmo 5.

***Linguistic Similarity* (*LS*):** calcula la similitud 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 similitud de acuerdo al número común de *qgramas* entre las etiquetas (Angell, y otros, 1983). El algoritmo *Check Synonym* utiliza el diccionario lingüístico WordNet (Miller, 1995), para identificar sinónimos, mientras el algoritmo C*heck Abbreviation* usa un diccionario de abreviaciones adecuado al dominio de aplicación. Si todos los algoritmos entregan un valor de 1, existe una coincidencia exacta entre las etiquetas, si entregan un valor de 0 no hay similitud entre las palabras. Si los valores entregados por *Ngram* y *CheckAbbreviation* son iguales a 0 y el valor de *Check synonym* está entre 0 y 1, el valor total de la similitud es igual a *Check Synonym.* Finalmente, si los tres algoritmos arrojan un valor entre 0 y 1, la similitud lingüística es el promedio de los tres. La función LS es descrita en el Algoritmo 5 (Corrales, 2008).

Algoritmo 5. Función Linguistic Similarity (LS).

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

Donde: m1 = Sim(NGRAM), m2 = Sim(Synonym Matching), m3 = Sim(Abbreviation Expansion)

3.1.4. Ejemplo de Matching de Actividades Básicas.

En esta sección se ejemplifica el emparejamiento a nivel atómico, mediante la comparación de actividades básicas contenidas en dos procesos BPEL para reservas hoteleras.

- Supongamos que el primer proceso BPEL tiene las siguientes actividades básicas: en primer lugar, el cliente debe seleccionar el hotel, actividad *SolicitudReserva* (tipo: *Receive*). Entonces, el mensaje, ya sea *MostrarCatalogo* o *MostrarDisponibilidad*, es enviado a través de la actividad *InformaciónHoteles* (tipo: *Pick*). Luego, las actividades tipo *Invoke*: *SolicitarCatalogo* o *SolicitarDisponibilidad* son ejecutadas respectivamente. Después, una confirmación (*ConfirmaciónUsuario*, tipo: *Reply*) con la

información de reserva es enviada al usuario. Por último, el servicio de reservas hoteleras espera para el pago de la tarjeta de crédito *PagoCC* (tipo: *Receive*).

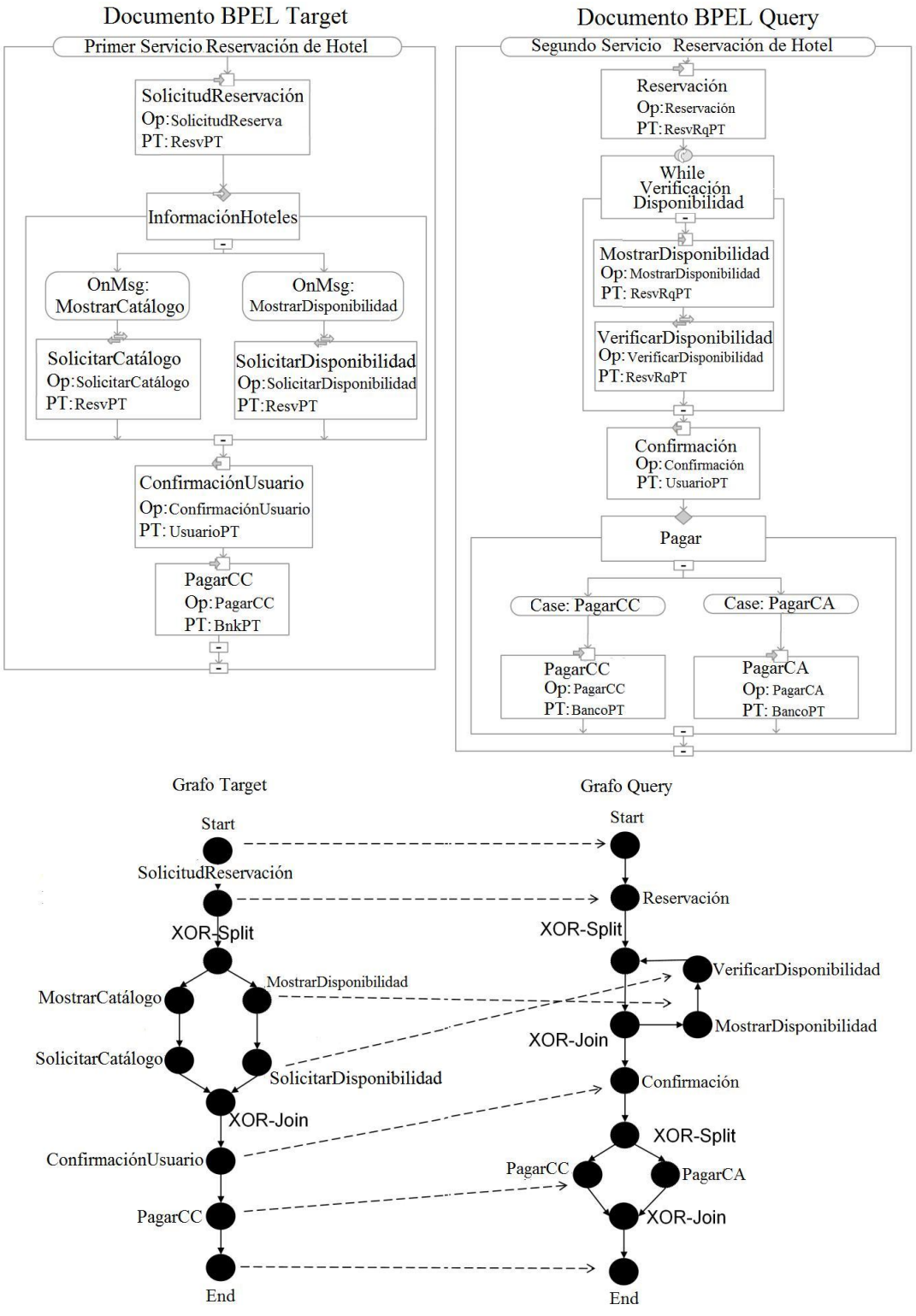


Figura 8. Ejemplo de Emparejamiento de Actividades Básicas BPEL

- El modelo del segundo proceso BPEL tiene la siguiente secuencia de actividades: en primer lugar, el cliente debe especificar sus preferencias de reserva, actividad *Reservación* (tipo: *Receive*). Luego, el servicio de reservas hoteleras recibe la fecha de reservación del cliente (*MostrarDisponibilidad* tipo: *Receive*) y verifica la disponibilidad de hoteles (*VerificarDisponibilidad* tipo: *Invoke*); si no hay habitaciones disponibles para las fechas propuestas, las dos últimas operaciones se repiten hasta encontrar habitaciones disponibles. A continuación, una confirmación (*Confirmación* tipo: *Reply*) se envía al usuario. Por último, el servicio de reservas hoteleras requiere que el cliente pague, actividad *Pago* (tipo: *Switch*), ya sea con tarjeta de crédito (*PagarCC* tipo: *Receive*) o por medio de cheque, actividad *PagarCA* (tipo: *Receive*).

- En primer lugar las descripciones BPEL de los dos procesos deben ser convertidas en sus equivalentes en grafos, transformando el emparejamiento de actividades básicas BPEL a un emparejamiento de los nodos que componen los grafos Query y Target (Figura 8). Luego, los nodos de cada grafo son obtenidos y organizados mediante la función *ClassifierNodes.* Una vez clasificados los nodos por tipo de actividad, la función *BasicActivityMatch* es invocada para compararlos. Para cada pareja de nodos actividad, la función *BasicActivityMatch* verifica inicialmente la similitud de sus operaciones. Por ejemplo, al comparar las actividades *Reservación* y *SolicitudReserva* (tipo: *Receive*), se determina la correspondencia entre las operaciones de las actividades utilizando la función de Similitud Lingüística *LS*.

- Como existe una similitud entre ellos, la función LS calcula la similitud entre los PortTypes (portType: ResvPT y portType: ResvRqPT) y los PartnerLinks, y finalmente, la distancia para cada pareja de nodos del mismo tipo de actividad es retornada. Para éste ejemplo, se ha considerado que la similitud de la operación (wop), tiene la misma importancia que las similitudes del portType (wpt) y partnerLink (wpk). En la Figura 8, se presenta la relación entre los nodos que tuvieron el máximo valor de similitud.

- En conclusión, el resultado del emparejamiento mostrará que los dos grafos tienen algunas actividades en común (*Start*, *SolicitudReserva*, *MostrarDisponibilidad*, *SolicitarDisponibilidad*, *ConfirmacionUsuario*, *Pagar* y *End*), pero las actividades *MostrarDisponibilidad* y *VerificarDisponibilidad* del grafo *Target* tienen un valor de similitud menor.

3.2. Personalización

Como se expuso en el estado del arte, la personalización es el proceso de adaptación automática del contenido de la información, su estructura y presentación para un usuario determinado (Zhu, et al., 2008). Los sistemas personalizados dan respuestas adaptadas a peticiones individuales (directas o indirectas), siendo capaces de acceder y seleccionar datos, conocer al usuario y aprender de él, y establecer la pertinencia de la información de acuerdo a las necesidades del mismo. Para conocer al usuario se deben obtener datos sobre su comportamiento, estado, gustos, etc.; esta información ha sido denominada perfil, contexto o preferencias de usuario en diferentes estudios (Asfari, y otros, 2009)(Barreiro, y otros, 2008). Por lo expuesto, en el presente trabajo de grado la personalización es soportada mediante la gestión del contexto y los perfiles de usuario. Características descritas a continuación.

3.1.5. Gestión del Contexto

(Steller, et al., 2009)y (Doulkeridis, et al., 2006)proponen una técnica de descubrimiento de servicios basada en significados semánticos e información del contexto, mejorando considerablemente el resultado de la búsqueda de los servicios solicitados. En estos trabajos se aprecia la utilidad de un modelo que utilice el "contexto" para enriquecer el proceso de matching durante el descubrimiento de servicios en ambientes pervasivos. En (Steller, et al., 2006), la gestión del contexto se maneja mediante atributos dinámicos que mantienen la información de contexto, salvo por su valor, el cual no es determinado hasta ser necesitado. Dichos atributos son establecidos por las siguientes consideraciones: *restricciones del contexto del solicitante* y *requerimientos del contexto del servicio*. Éstas variables pueden ser soportadas por los Repositorios de Servicios y Dispositivos definidos en el estado del arte. A continuación se describe la forma como dichos repositorios soportan la gestión del contexto en el presente proyecto de grado.

3.1.5.1. Requerimientos de Contexto del Servicio.

El Repositorio de Servicios, soportado por el trabajo presentado en (Vanhatalo, et al., 2006), permite almacenar documentos BPEL y otro tipo de archivos basados en el estándar XML. Estos archivos enriquecen el proceso de negocio con características de contexto. Por lo tanto, es necesario definir un metadato XML que describa el contexto en el cual el servicio puede ser consumido. Dicha información es entregada por los proveedores o desarrolladores de servicios. Cumpliendo así con unas de las variables de contexto establecidas en (Steller, et al., 2006) denominada: *requerimientos de contexto del servicio.*

Para que el *Repositorio de Servicios* soporte la característica de contexto definida anteriormente, es necesario extender su funcionalidad mediante un nuevo modelo EMF (*Eclipse Modeling Framework*) que permita describir las restricciones del servicio especificadas por el proveedor. Para cumplir con Este requerimiento, es conveniente definir un nuevo *meta-dato* que permita expresar/representar las restricciones o requerimientos del contexto del servicio. El *meta-dato* es denominado *features.context* y su estructura es presentada en la Figura 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<emfcontext  xmi:version="2.0"  xmlns:xmi="http://www.omg.org/XMI"
xmlns:emfcontext="http:///com/ibm/bpia/repository/extensions/emfcontext.ecore" process="Holidays">
  <networkaccesses>
        <networkaccess>bluetooth</networkaccess>
        <networkaccess>wifi</networkaccess>
        <networkaccess>gsm</networkaccess>
        <networkaccess>gprs</networkaccess>
  </networkaccesses>
  <display>
        <resolution_width>320</resolution_width>
        <resolution_height>480</resolution_height>
        <columns>11</columns>
        <max_image_width>320</max_image_width>
        <max_image_height>36</max_image_height>
        <rows>6</rows>
  </display>
</emfcontext>
```

Figura 9. Meta-Dato de los requerimientos de Contexto del Servicio.

El archivo *features.context* contiene el nombre del servicio (*Holidays*), una lista de los tipos de red soportados(bluetooth, wifi, gsm, gprs) y también una lista de la características de presentación/visualización que el dispositivo debe soportar para poder consumir los servicios invocados. El procedimiento para extender el repositorio con nuevos modelos EMF es presentado en el **ANEXO C.** El proveedor de servicios publica tres tipos de documentos que

describen sus servicios: BPEL, WSDL y *features.context*, archivos almacenados en el repositorio de servicios. Cabe resaltar que múltiples características se pueden definir en el Meta-Dato *features.context.* Sin embargo para el presente proyecto se decidió considerar solo el tipo de acceso como característica de contexto. Ya que el estudio y valoración de un solo parámetro permite establecer su funcionalidad. Sin embargo la posibilidad de tener en cuenta más parámetros está abierta.

3.1.5.2. Restricciones del Contexto del Usuario

El Repositorio de Dispositivos soporta la segunda variable, que corresponde a la descripción de capacidades de los dispositivos que solicitan los servicios, denominada: *restricciones del contexto del solicitante*, que son las características de los dispositivos clientes tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. Información recuperada por medio del repositorio UAProf y la librería WURFL.

Según (Guerrero, et al., 2010)las restricciones del contexto del solicitante, se definen como cualquier información que pueda ser utilizada para caracterizar una entidad, donde una entidad puede ser una persona o cualquier objeto que es considerado relevante para la interacción entre un usuario y una aplicación. Propone tres dimensiones que permiten definir un meta-modelo de contexto del usuario, detalladas a continuación y representadas en la Figura 10.

- Dimensión espacial: esta dimensión contiene todos los parámetros que están asociados con información geográfica y espacial, como son la longitud y latitud desde la cual un usuario accede al servicio.
- Dimensión temporal: contiene información sobre el momento (tiempo) y la fecha en la cual se hace una petición al sistema o es invocado el servicio.
- Dimensión de los datos del equipo: la información del equipo como software soportado e instalado, hardware, tipo de sistema operativo, son útiles para procesos de adaptación de contenido, estos parámetros abarcan además, la información contenida en diferentes estándares utilizados para éste propósito como el CC/PP.

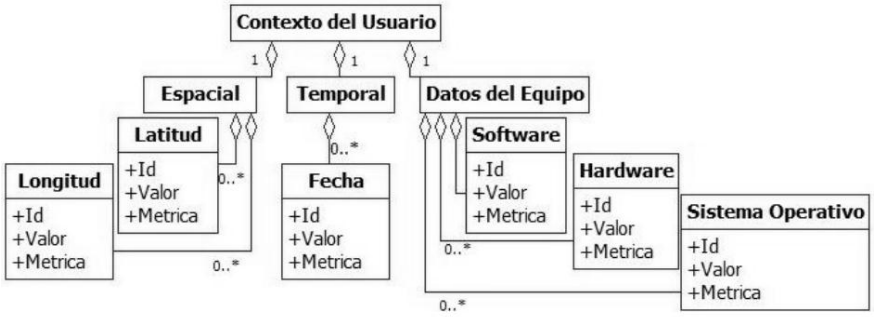


Figura 10. Meta-Modelo de Contexto del Usuario

Una vez identificadas las variables que intervienen en la gestión del contexto, procedemos a definir el algoritmo que constituye un método para resolver el problema del *Contexto*, tanto del *Usuario* como del proveedor de servicios en el proceso de Descubrimiento. La función denominada *CheckDeliveryContext* constituye el soluciona dicho problema, a continuación se describe el algoritmo definido.

Esta función toma la lista del ranking de los servicios obtenidos en la fase de emparejamiento de actividades básicas y verifica los requerimientos de contexto de los servicios recuperados en el *Repositorio de Servicios*. Las restricciones de contexto del usuario, son encontradas a través del *Repositorio de Dispositivos* mediante las cabeceras HTTP, UAProf y WURFL. Así, las características de los dispositivos, son relacionadas con las restricciones del proveedor de servicios (protocolos de acceso y redes). En primer lugar, el perfil del dispositivo es obtenido desde el *Repositorio de Dispositivos*. Luego, el descriptor *features.context*es recuperado para cada servicio contenido en el *Repositorio*. Posteriormente, la función verifica que el dispositivo es compatible con las tecnologías de acceso definidas por el proveedor de servicios en el archivo descriptor *features*.*context*. El Algoritmo 6 describe la función *checkDeliveryContext*.

Algoritmo 6. Función Verificar Contexto - CheckDeliveryContext

```
INPUTS: (listRankedServices)
OUTPUT: rankedFilteredServices

Device Profile deviceProfile=  lookupDeviceProfile(deviceURL)
for each node in listRankedServices do
    EmfContext features.context = lookupFeatures.context (node.URI)
    for each networkaccess in features.context do
        if deviceProfile contains networkaccess (network access supported) then
            Add node to rankedFilteredServices
            break for
        end if
    end for
end for

Return rankedFilteredServices
```

3.1.6. Perfil de Usuario

De (Guerrero, et al., 2010) se tiene que los diferentes conjuntos de datos que conforman el Perfil de usuario y que se han denominado Dimensiones del Perfil de usuario son: datos personales y dominio de intereses. En laFigura 11, se presenta el meta-modelo general del Perfil de usuario propuesto. A continuación se explica cada una de estas dimensiones.

- Dominio de interés: la información de esta dimensión obtenida de un usuario depende del ámbito de aplicación. Varios estudios utilizan diferentes métodos de obtención y manipulación de la información, dependiendo de la aplicación: Web mining: (Tocarruncho Tocarruncho, y otros, 2007), clustering: (Zhang, y otros, 2009), vectores de registros (logs) de aplicaciones (Abbar, et al., 2008), etc., cada uno de estos mecanismos, genera un conjunto de parámetros con sus posibles valores para un dominio de interés dado. La definición de estos parámetros y valores no se establece en éste trabajo, debido al alto nivel de análisis y la desvinculación a un ámbito específico.

- Datos personales: la dimensión de los datos personales se divide en dos categorías, la de los Datos de Identificación y los Datos Demográficos, esto se establece debido a los requerimientos de protección de la identificación del usuario.

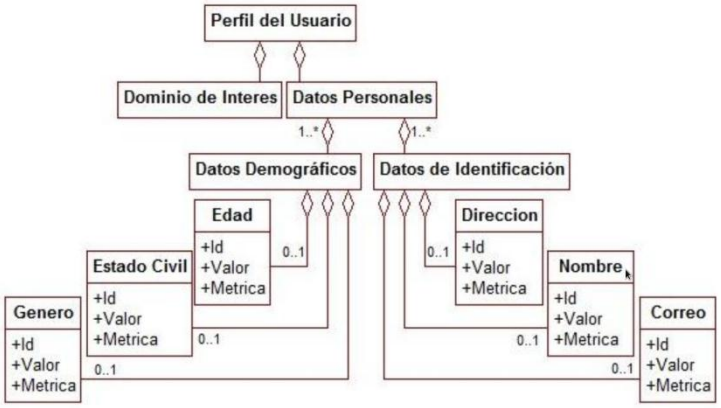


Figura 11. Meta-modelo del perfil de usuario

Estas descripciones se almacenan en el Repositorio de Usuarios. A partir de esta descripción, el sistema de descubrimiento de servicios realiza un proceso de sugerencia de servicios. Este proceso tiene como finalidad recomendar un grupo de servicios ya utilizados por el usuario y/o personas con perfiles similares, y así disminuir el tiempo que toma el descubrimiento realizado sobre todo el repositorio de servicios. Este procedimiento de búsqueda y recomendación, precede a la fase de descarte realizada por la función de verificación de contexto, expuesta en la sección anterior. El primer paso en la fase de sugerencia es encontrar usuarios similares, para después utilizar los servicios consultados y consumidos por ellos.

La función *FindSimilarProfiles*, es la encargada de organizar una búsqueda sobre los perfiles contenidos en el repositorio de usuario (ver Algoritmo 7, definido en este proyecto). Los perfiles de usuario están definidos de acuerdo al meta-modelo de perfil de usuario ya presentado. Para determinar la distancia entre los perfiles, la función *FindSimilarProfiles* hace uso de la función *basicProfileMatch* (ver Algoritmo 8, definido en este proyecto).

Algoritmo 7. Function FindSimilarProfiles

INPUTS: (*queryProfile*)
OUTPUT: similarProfilesList
BEGIN
List Profiles *existingProfiles* = lookupServiceRepository(non-operational information)
for each $profile_i$ **in** *existingProfiles* **do**
DistanceProfile*distance* = basicProfileMatch($profile_i$, *queryProfile*)
    if *distance* < 1 **then**
        Add ($profile_i$, *distance*) to similarProfilesList
    end if
end for
return similarProfilesList.

La función *BasicProfileMatch* calcula la distancia entre una pareja de perfiles. Este cálculo se realiza sobre valores de características que describen el perfil de usuario. Debido al aumento en procesamiento y tiempo, se descartó la posibilidad de utilizar razonadores en esta comparación. Por eso la distancia es calculada utilizando un diccionario léxico como WordNet, haciendo éste procedimiento similar al matching básico de actividades. Esta función comienza evaluando la similitud de las edades *SimAge,* de la misma manera luego evalúa la similitud entre el estado civil *SimCS*y la similitud de género *SimG,* continua con cálculo de la similitud de valores de la lista de Dominios de interés *SimDoIij* para estimar la similitud SimDoI general, por último se calcula distancia entre los dos perfiles, *DistanceProfile*. Los pesos *Wage*, *Wcs,Wg y* W_{DoI} indican la contribución de la similitud de los atributos DoIlisti, Agei, CSi, y Gia la similitud de los perfiles ($0 \leq Wage \leq 1$, $0 \leq Wcs \leq 1$, $0 \leq Wg \leq 1$ y $0 \leq W_{DoI} \leq 1$). Para calcular la similitud de los valores de *DoI* (Dominio de Interés) se emplea la función *LinguisticSimilarity* (*LS*), descrita en el Algoritmo 5.

Algoritmo 8. Function BasicProfileMatch

INPUTS ($Profile_i$, $Profile_j$) $Profile_i$: Struct ($DoIlist_i$, Age_i, CS_i, G_i), $Profile_j$: Struct ($DoIlist_j$, Age_j, CS_j, G_j)
OUTPUT: DistanceProfile
BEGIN
Calculate Age Similarity $SimAge = 1-\{[|Age_i - Age_j|/[(Age_i+Age_j)/2]\}$
SimCS = 0, SimG = 0, SimDoI = 0
if $CS_i = CS_j$ **then**
    SimCS =1
end if
if $G_i = G_j$ **then**
    $SimG$ =1
end if
for each value v_i**in** $doilist_i$ **do**
  for each value v_j**in** $doilist_j$ **do**
  Calculate DoI_{ij} Similarity $SimDoI_{ij} = LS(v_i,v_j)$
    If $SimDoI_{ij}$>SimDoI **then**
      SimDoI=$SimDoI_{ij}$
    end if
  end for
end for

$$\text{DistanceProfile} = 1 - \frac{w_{age} * \text{SimAge} + w_{cs} * \text{SimCS} + w_g * \text{SimG} + w_{DoI} * \text{SimDoI}}{w_{age} + w_{cs} + w_g + w_{DoI}}$$

En el Algoritmo 9 (definido en este proyecto), se expone el procedimiento para conseguir los servicios sugeridos antes del refinamiento de la búsqueda por parte de la función de verificación de contexto. En él se observa dos funciones, que evidencian la utilidad del perfil de usuario en el proceso de sugerencia de servicios. Éstas son: *listRankedServicesQueried* y *consumedNodes*. Internamente estas dos funciones utilizan a la función *findSimilarProfiles*.

El Algoritmo 9, empieza verificando si el nodo de consulta ya ha sido previamente solicitado, en caso afirmativo, recupera del repositorio de usuarios la información de los servicios retornados en esa oportunidad. En éste punto, es importante distinguir la diferencia entre un servicio de consulta y un servicio sugerido. El servicio de consulta es entregado por el

usuario, contiene una descripción de sus requerimientos, y puede ser real o no. En cambio los servicios sugeridos, obligatoriamente deben poseer una implementación. La importancia de los servicios previamente consultados radica en que si bien no poseen una implementación real, tienen relacionada una lista de servicios reales que les fueron sugeridos. Pero si el servicio de consulta no ha sido solicitado en ocasiones anteriores, se realiza una primera búsqueda sobre aquellos nodos que han sido consumidos por el usuario y/o usuarios con perfiles similares. Estos nodos son organizados con base en su distancia al nodo de consulta *queryNode*. Una vez terminada esta organización, la función *badSuggest* verifica que los nodos de *listRankedServices* cumplan con unas condiciones (superando un umbral y un número de servicios, mínimos requeridos por el usuario) para ser entregados al usuario como los más aproximados a sus requerimientos. Si ésta lista no cumple con estas condiciones, la búsqueda comienza desde cero, analizando todo el repositorio de servicios

Algoritmo 9. Servicios Sugeridos - SuggestServices

```
INPUTS: (queryNode, usrProfile)
OUTPUT: listRankedServices
BEGIN
if searchServicesPreviouslyConsulted(queryNode) then
    Get listRankedServices = listRankedServicesQueried(queryNode)
else
    Get List NodessuggestServices = consumedNodes(usrProfile)
    for each node_i in suggestService do
    CalculteDistanceNodedistance = basicActivityMatch(node_i, queryNode)
        if distance < 1 then
            Add ( nodei, distance) to listRankedServices order by distance
        end if
    end for
end if
if badSuggest(listRankedServices) then
    listRankedServices = null
    Get List Nodes candidateService = lookupServiceRepository(non-operational information)
    for each node_i in candidateService do
    Calculate DistanceNodedistance = basicActivityMatch(node_i, queryNode)
        if distance < 1 then
            Add ( node_i, distance) to listRankedServices order by distance
        end if
    end for
end if
Return listRankedServices
```

3.1.7. Ejemplo de Personalización (Descubrimiento de Servicios en Ambientes Ubicuos)

En esta sección, se expone el proceso de descubrimiento de servicios en ambientes ubicuos por medio de un ejemplo que muestra un usuario con la firme intención de comprar un libro electrónico, utilizando un sistema de descubrimiento de servicios en ambientes ubicuos, (SDSAU) para tal fin. Éste sistema contiene información sobre el perfil de sus clientes,

también un módulo de obtención de características del equipo que accede al sistema y un conjunto de servicios publicados. El proceso de SDSAU está conformado por cuatro fases:

1. Búsqueda de perfiles similares, Algoritmo 7
2. Sugerencia de Servicios, Algoritmo 9
3. Emparejamiento y organización según el grado de relevancia, Algoritmo 4
4. Clasificación y Descarte por contexto, Algoritmo 6.

Para el cliente son definidos 3 casos: Afortunado (A), Normal (N) e Infortunado (I).

Caso Afortunado:
Resulta que el cliente (C) ingresa a la plataforma, el sistema identifica a un IPAD como dispositivo utilizado. C consulta “Del Amor y Otros Demonios Gabriel García Márquez”, los clientes de SDSAU, para éste caso, son gente muy aficionada a la lectura y por eso SDSAU provee un variado surtido de Libros electrónicos.

Fase 1: en la búsqueda de perfiles, el sistema da como resultado perfiles de gran similitud muy compatibles con el perfil de C.

Fase 2: la sugerencia de servicios fue todo un éxito ya que en la primera etapa de esta fase se encontró que algunos usuarios recomendados hicieron una búsqueda igual a la que realizó C.

Fase 3: no fue necesaria ya que la fase 2 entregó una lista organizada de servicios útiles.

Fase 4: todos los servicios sugeridos fueron consumibles por el dispositivo, debido a sus excelentes capacidades.

Conclusión: en éste caso se aprecia de manera especial el papel de la fase de sugerencia de servicios, mostrando como ésta es capaz de reducir los tiempos de la búsqueda y eliminar el procesamiento de la fase 3.

Caso Normal:

C ingresa a la plataforma, el sistema detecta que lo hace desde un Samsung STAR S5230. C consulta “Del Amor y Otros Demonios Gabriel García Márquez”. En éste caso los clientes de SDSAU son lectores esporádicos, así que SDSAU provee un surtido moderado de Libros electrónicos.

Fase 1: el sistema retornó una cantidad, no muy grande, de perfiles aproximados a las características de C.

Fase 2: no se encontró una consulta igual a la realizada por C; por eso retorna los servicios que han sido consumidos anteriormente por los usuarios sugeridos.

Fase 3: esta fase comparó la solicitud de C con los servicios sugeridos, obteniendo una lista de servicios similares, que cumplió las condiciones de calidad del usuario.

Fase 4: de la lista recomendada por la fase 3, la mitad de los servicios requerían mayores características de procesamiento y tamaño de pantalla. Por eso, al usuario fue retornada la otra mitad de servicios, aptos para el consumo desde su dispositivo.

Conclusión: En éste caso, tal como en el anterior, se rescata el papel de la fase 2, al evitar que la fase 3 realice una búsqueda sobre toda la base de servicios ofrecidos, esto disminuyó el tiempo para la entrega de una respuesta, y retornó los servicios consumidos más aproximados a los requerimientos de usuario.

Caso Infortunado:
C ingresa a la plataforma, el sistema identifica un Sony Ericsson W300 como dispositivo de acceso. C consulta "Del Amor y Otros Demonios Gabriel García Márquez". En éste caso los clientes de SDSAU son gente no habituada a la lectura y por eso SDSAU provee un escaso surtido de Libros electrónicos.

Fase 1: en esta fase no se encontró perfiles similares al de C.

Fase 2: al no tener usuarios sugeridos esta etapa retornó la información de los servicios antes consumidos por C.

Fase 3: comparó los servicios antes consumidos con la actual solicitud pero la lista resultado no superó las características de calidad requeridas por el usuario. Así, se procedió a buscar sobre todos los servicios publicados en SDSAU, esta búsqueda dio como resultado una pequeña lista de servicios recomendados.

Fase 4: en esta fase se descartó el 90% de los servicios encontrados, dado que el dispositivo de C no ofrece las características necesarias para consumirlos.

Conclusión: en éste caso, C no obtuvo buenos resultados en su búsqueda, debido a que su entorno no ofrecía los servicios que él requería, además los usuarios registrados en el sistema tenían perfiles muy distintos al suyo, lo cual incrementó el tiempo de búsqueda su servicio.

En conclusión, la satisfacción del usuario en el proceso de descubrimiento en ambientes ubicuos, depende tanto de características técnicas como de las características de su entorno. Mostrando así, que el desempeño de éste tipo de búsqueda, es afectado directamente por las situaciones que rodeen al usuario, siendo estas tan diversas, como perfiles de usuario puedan existir.

3.3. Resumen

En éste capítulo se presentó los diferentes algoritmos que intervienen en el Descubrimiento de servicios en ambientes ubicuos. Inicialmente se expuso el emparejamiento de actividades básicas BPEL, que opera sobre un modelo de grafos. Éste proceso se realizó utilizando una comparación lingüística de los parámetros de las actividades básicas BPEL. Seguido, se presentó la personalización del proceso de descubrimiento de servicios, abordada desde la gestión del contexto y el perfil de usuario. En esta sección se evidenció la utilidad de los repositorios de Servicios, Usuarios y Dispositivos, como herramientas de soporte al proceso de descubrimiento de servicios en ambientes ubicuos. Por último, se presentó los algoritmos involucrados en el descubrimiento de servicios en ambientes ubicuos y un ejemplo de su aplicación.

Capítulo IV

ARQUITECTURA PLATAFORMA U-ServiceMatch

En este capítulo se presentan los resultados del proceso de ingeniería llevado a cabo para el desarrollo de la arquitectura que soporta la plataforma propuesta, **U-ServiceMatch**.

U-ServiceMatch, permite el descubrimiento de servicios en ambientes de computación ubicua, está compuesta por un módulo de emparejamiento a nivel atómico de procesos BPEL, es decir, un emparejamiento de las actividades básicas que componen dicho proceso, y tres repositorios que permiten su aplicación en ambientes de computación ubicua. Dado a las características y dificultades que éste tipo de entornos demanda, los repositorios deben soportar: el almacenamiento de las descripciones de los servicios, la identificación del contexto en el cual el servicio será consumido y las preferencias de usuario, que son características fundamentales en un contexto ubicuo.

Para el desarrollo de los componentes software que conforman la plataforma propuesta, se utilizó la metodología sugerida en el Modelo de Construcción de Soluciones (Serrano, 05). La cual plantea el desarrollo de un prototipo de valoración considerando las siguientes fases: Establecimiento de responsabilidades, Descripción de la solución, Definición de la Arquitectura, Implementación y Puesta a Punto (Anexo A) y Evaluación (Capítulo V). Para este proyecto, la etapa de Implementación y Puesta a Punto, se desarrolló en iteraciones incrementales, incluyendo actividades de Verificación y Validación al final de cada iteración.

Se plantearon las siguientes iteraciones:

- **Iteración 1:** Implementación del prototipo de Emparejamiento de Actividades BPEL, Integración del Repositorio de Servicios con el módulo de Emparejamiento (ServiceMatch), Verificación y Validación.

- **Iteración 2:** Definición del metadato que describe los requerimientos de contexto del servicio, Implementación del mecanismo de consulta de los metadatos de contexto. Adaptación del prototipo *ServiceMatch* al contexto, Implementación del Repositorio de Capacidades de Dispositivos, Integración del prototipo *ServiceMatch* con el Repositorio de Capacidades de Dispositivos (*SeMatch-Context*), Verificación y Validación.

- **Iteración 3:** Implementación del Repositorio de Usuarios, Integración del prototipo *SeMatch-Context* con el Repositorio de Usuarios (U-ServiceMatch), Verificación y Validación.

4.1. Establecimiento de responsabilidades

En esta etapa se determina la viabilidad y alcance del proyecto, mediante las bases conceptuales, la investigación de alternativas y demás actividades que permiten estructurar el presente trabajo de investigación. Ésta fase se compone de las siguientes actividades: Consideraciones Iníciales, Captura y Análisis de Requisitos Funcionales e Identificación de Casos de Uso.

4.1.1. Consideraciones Iníciales

A partir de la investigación documental realizada, se presentó la información acerca de las bases teóricas y los trabajos relacionados (capítulo 2), de los cuales, surgieron diferentes consideraciones a tener en cuenta en el desarrollo de la plataforma. Éstas son descritas a continuación:

4.1.1.1. Descripción Formal de Procesos de Negocio y Emparejamiento de Servicios

En el presente proyecto de grado, se escogió BPEL como lenguaje para describir los servicios disponibles en la red ubicua. Sin embargo, manipular dichas descripciones, basadas en el esquema XML, no es una tarea fácil. Por ello es necesario un modelo formal adecuado, que permita representar los procesos de negocio para facilitar su posterior tratamiento. En (Corrales, et al., 2006) (Grigori, et al., 2005)los autores plantean la trasformación de la descripción BPEL a Grafos, con el objetivo de trasladar el problema de emparejamiento de procesos de negocios, a un problema matemático de emparejamiento de grafos, a través de reglas formales ya establecidas. Por esta razón, es necesario basar nuestro enfoque sobre la recuperación de servicios, en un emparejamiento a nivel atómico de Grafos (Capítulo III). Dado que un proceso BPEL se puede representar como un Grafo, cuyos nodos son las actividades y las aristas tienen las condiciones de transición que expresan las dependencias del control de flujo entre las actividades.

4.1.1.2. Contexto

Debido al incremento en las capacidades de movilidad, el entorno del usuario, la localización y los objetos alrededor del mismo son más dinámicos. Esto obliga a considerar el contexto durante el proceso de descubrimiento. Exigiendo el uso de información implícita que relacione tanto las limitaciones/restricciones del solicitante, como los requerimientos del proveedor de servicios, lo cual afecta la calidad y relevancia de los resultados retornados por los mecanismos de recuperación de servicios (Doulkeridis, et al., 2003) (Sellami, et al., 2009). (Steller, et al., 2009)Y (Doulkeridis, et al., 2006) proponen una técnica de descubrimiento de servicios basada en significados semánticos e información del contexto, mejorando considerablemente el resultado de la búsqueda de los servicios solicitados. En estos trabajos, se aprecia la utilidad de un modelo que utilice el "contexto" para enriquecer el proceso de matching durante el descubrimiento de servicios en ambientes ubicuos. En (Steller, et al., 2006), la gestión del contexto se maneja mediante atributos dinámicos que mantienen la información de contexto, salvo por su valor, el cual no es determinado hasta ser necesitado. Dichos atributos son establecidos por las siguientes consideraciones: limitaciones del contexto del solicitante y requerimientos del contexto del servicio.

De los anteriores trabajos, se concluye que es conveniente implementar componentes que gestionen el contexto durante el proceso de descubrimiento. Las siguientes consideraciones permiten lograr dicho objetivo:

Gestión del Contexto:

La gestión del contexto puede ser soportada por los Repositorios de Servicio y de Capacidades de Dispositivos definidos en el estado del arte.

- El Repositorio de Servicios, soportado por el trabajo presentado en (Vanhatalo, et al., 2006), permite almacenar documentos BPEL y otro tipo de archivos basados en el estándar XML. Estos archivos enriquecen el proceso de negocio con características de contexto. Por lo tanto, es necesario definir un metadato XML que describa el contexto en el cual el servicio puede ser consumido. Dicha información es entregada por los proveedores o desarrolladores de servicios. Cumpliendo así con unas de las variables de contexto establecidas en (Steller, et al., 2006) denominada: requerimientos de contexto del servicio.

- La segunda variable corresponde a la descripción de capacidades de los dispositivos que solicitan los servicios, denominada: restricciones del contexto del solicitante, que son las características de los dispositivos clientes tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. Esta información es recuperada por medio del repositorio UAProf y la librería WURFL.

4.1.1.3. Personalización

La personalización es la capacidad de adaptar un servicio o contenido a un usuario, en función de sus características, preferencias personales o información proporcionada previamente. Por ello, es conveniente implementar un repositorio que permita almacenar detalles como preferencias explicitas provistas por los usuarios (ej. Lenguaje nativo) o datos implícitos capturados dinámicamente tales como historial de servicios consumidos o patrones de uso.

En el *Repositorio de Usuarios,* se registran las preferencias que permiten crear el *perfil de usuario*. Esto puede ser actualizado posteriormente, con base en el comportamiento del mismo, por ejemplo, servicios consumidos o historial y patrones de uso. Las preferencias de usuario proporcionan un mecanismo de personalización, que puede ser visto como parte del contexto global, y permite que el descubrimiento de servicios se ajuste mejor a las exigencias del usuario.

4.1.2. Captura y Análisis de Requisitos Funcionales

4.1.2.1. Alcance del sistema

La plataforma U-ServiceMatch permite el descubrimiento de servicios en ambientes de computación ubicua. Su desarrollo ha seguido un proceso iterativo e incremental, obteniendo de cada iteración un prototipo de validación: *ServiceMatch*, *SeMatch-Context* y *Repositorio de Usuarios*. La integración del *Repositorio de Usuarios* con la aplicación *SeMatch-Context* constituye la plataforma *U-ServiceMatch*.
A continuación se presenta el alcance de cada uno de los componentes que conforman la plataforma *U-ServiceMatch*.

ServiceMatch

Es una aplicación de escritorio que 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 cuatro atributos: *Tipos de Actividad, Nombre de la operación PartnerLink* y *PortType*. Esta aplicación emplea una representación formal de grafos para los requerimientos del usuario y los servicios almacenados en el repositorio, así el problema de

emparejamiento de actividades, se transforma en un problema de emparejamiento de nodos. Dicho emparejamiento se realiza a nivel atómico, comparando las actividades básicas que conforman el requisito del cliente contra las actividades contenidas en el repositorio de servicios.
Los resultados del emparejamiento se pueden visualizar de manera textual mediante la interfaz de la aplicación *ServiceMatch*, donde se muestra, el tipo de actividad de los nodos emparejados, el nodo de consulta, la colección de los nodos del repositorio y el resultado, presentado como una lista ordenada de los nodos más relevantes recuperados para un top K, junto con el valor de similitud. Además, la interfaz permite al usuario fijar unos pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType* a la similitud de las actividades.

SeMatch-Context

Es una aplicación Web que relaciona el emparejamiento a nivel atómico de actividades básicas con el contexto, tanto de las restricciones del solicitante, como de los requerimientos del servicio. El *SeMatch-Context* está compuesto por dos subsistemas: el *ServiceMatch* que debe ser adaptado al contexto y el *Repositorio de Capacidades de Dispositivos*, que debe permitir almacenar las características de los dispositivos móviles tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc.
La adaptación del *ServiceMatch* se da gracias a la flexibilidad del repositorio de procesos que lo compone. Para esto, es necesario definir un metadato que describa los requerimientos de contexto del servicio, ya que el *Repositorio de Servicios* permite almacenar otro tipo de archivos basados en el estándar XML.

Esta aplicación al igual que el *ServiceMatch* toma como entradas dos nodos, que representan actividades básicas de BPEL (*receive, invoke, reply*), y calcula la distancia semántica entre ellos. Sin embargo, a diferencia del *ServiceMatch*, los nodos en el *SeMatch-Context* poseen un atributo adicional que relaciona el contexto: *Tipo de Acceso*. Por lo tanto, una vez terminado el proceso de recuperación, la aplicación verifica si las actividades recuperadas pueden ser consumidas en el contexto del usuario.

Los resultados del emparejamiento se pueden visualizar de manera textual en el browser del cliente móvil, mediante la interfaz de la aplicación *SeMatch-Context*, donde se muestra el tipo de actividad de los nodos emparejados, el nodo de consulta, la colección de los nodos del repositorio de servicios y el resultado presentado como una lista ordenada de los nodos más relevantes recuperados para un top K, junto con el valor de similitud y el atributo de contexto.

Con respecto a los usuarios que interactúan con la plataforma, se pueden distinguir los siguientes:

- *Administrador del sistema*: representa a la persona encargada de administrar la plataforma U-ServiceMatch, la cual gestiona las políticas de descubrimiento y la información de los usuarios registrados en el sistema.

- *Cliente Móvil*: representa al usuario con privilegios de uso de la plataforma, entre estos privilegios se encuentra: la consulta de los servicios consumidos, búsqueda de servicios y publicación de nuevos servicios.

- *Proveedor de Servicios*: representa la persona encargada de publicar sus procesos de negocio (BPEL) con la respectiva descripción del contexto en el *Repositorio de Servicios.*

Requisitos Funcionales del Sistema

Los requisitos funcionales identificados para la plataforma son los siguientes:

***Gestionar los usuarios de la plataforma*.** Éste requisito debe permitir registrar al usuario y sus preferencias, y con ello crear su perfil.

Publicar los documentos BPEL y archivo de contexto asociado. La plataforma debe brindar una interfaz que permita publicar los documentos BPEL y los requerimientos del contexto asociado a dicho proceso en el *Repositorio de Servicios*.

Realizar un descubrimiento a nivel atómico basado en las características de contexto y preferencias del usuario. La plataforma debe permitir descubrir los servicios más relevantes, mediante una fase de emparejamiento a nivel atómico de las actividades BPEL, considerando el contexto y las preferencias del usuario.

Entregar los servicios recuperados más relevantes. La plataforma debe entregar una lista de los servicios más adecuados encontrados, las cuales, una vez terminada la fase de emparejamiento de las actividades básicas que conforman el requisito del cliente contra las actividades contenidas en el *Repositorio de Servicios*.

Monitorear el comportamiento del usuario. A parte de los datos explícitos provistos por los usuarios al registrarse. La plataforma debe permitir obtener información implícita capturada dinámicamente, tal como servicios consumidos y patrones de uso.

Gestionar las políticas de descubrimiento. La plataforma debe permitir manejar las variables de las rutinas de control de actualización de la información, dentro del sistema y también la definición de los pesos de los parámetros, en el algoritmo de emparejamiento.

4.1.3. Identificación de Casos de Uso

En esta sección se presenta el diagrama de casos de uso de la aplicación software desarrollada en el presente trabajo de grado. Los casos de uso se identificaron a partir de las funcionalidades que ofrece el sistema. Los casos de uso en formato extendido se presentan en el **Anexo A**.

4.1.3.1. Diagrama de Casos de Uso de la Plataforma U-ServiceMatch

En la Figura 12, se presentan el diagrama de casos de uso para U-ServiceMatch, a continuación se realiza una breve descripción de ellos.

a) *Registrar Datos y Preferencias*: permite registrar información personal y preferencias de usuario, para posteriormente crear su perfil dentro del sistema.

b) *Publicar Servicios*: permite publicar los documentos BPEL y los requerimientos del contexto asociado a dicho proceso en el *Repositorio de Servicios*.

c) *Recuperar Servicios: e*ste caso de uso permite a un cliente móvil consultar y recuperar los servicios más relevantes que se ajusten a su petición, mediante una

fase emparejamiento a nivel atómico de actividades BPEL, considerando tanto las restricciones del contexto, como las preferencias del usuario.

d) *Obtener Comportamiento del Usuario:* este caso de uso permite a la plataforma obtener información del comportamiento del usuario capturada dinámicamente, tal como servicios consumidos y patrones de uso del sistema.

e) *Obtener Preferencias del Usuario*: para la fase de descubrimiento es conveniente considerar las preferencias de usuario, es por ello que éste caso de uso permite obtener el *perfil de usuario* para dicho proceso.

f) *Obtener Actividades de los procesos BPEL*: el proceso de descubrimiento se apoya de una fase de emparejamiento a nivel atómico, es decir, un emparejamiento de las actividades básicas de los procesos BPEL. Por esta razón, éste caso de uso permite obtener dichas actividades, tanto aquellas que conforman el requisito del cliente, como las actividades de los procesos contenidos en el *Repositorio de Servicios*.

g) *Registrar Comportamiento de Usuario*: una vez el proceso de recuperación de servicios ha sido ejecutado, el sistema registra tanto los servicios consumidos, como los que simplemente han sido consultados. Este caso de uso permite registrar dicho comportamiento en el *Repositorio de Usuarios* y actualizar el perfil de los usuarios.

h) *Emparejar Actividades Básicas*: éste caso de uso permite comparar dos nodos, que representan actividades básicas de BPEL (*receive, invoke, reply*), y calcular la distancia semántica entre ellos.

i) *Identificar el Contexto de Entrega*: éste caso de uso permite identificar las restricciones del solicitante y obtener los requerimientos del servicio. Información relevante, para el proceso de descubrimiento de servicios en entornos tan dinámicos como los ambientes ubicuos.

j) *Gestionar Sistema*: este caso de uso, le permite al usuario administrador ciertos privilegios a la hora de interactuar con la plataforma, tales como: gestionar toda la información de usuarios y las políticas de descubrimiento del sistema.

k) *Gestionar Información de Usuarios*: éste caso de uso, le permite al administrador del sistema gestionar los *Perfiles de Usuario*, como: crear, modificar y actualizar parámetros de los perfiles. Igualmente puede crear, consultar, editar y eliminar los diferentes usuarios que tienen acceso a la plataforma.

l) *Gestionar Políticas de Descubrimiento*: este caso de uso, le permite al administrador del sistema manejar las variables de las rutinas de control de actualización de la información dentro del sistema y también la definición de los pesos de los parámetros en el algoritmo de emparejamiento.

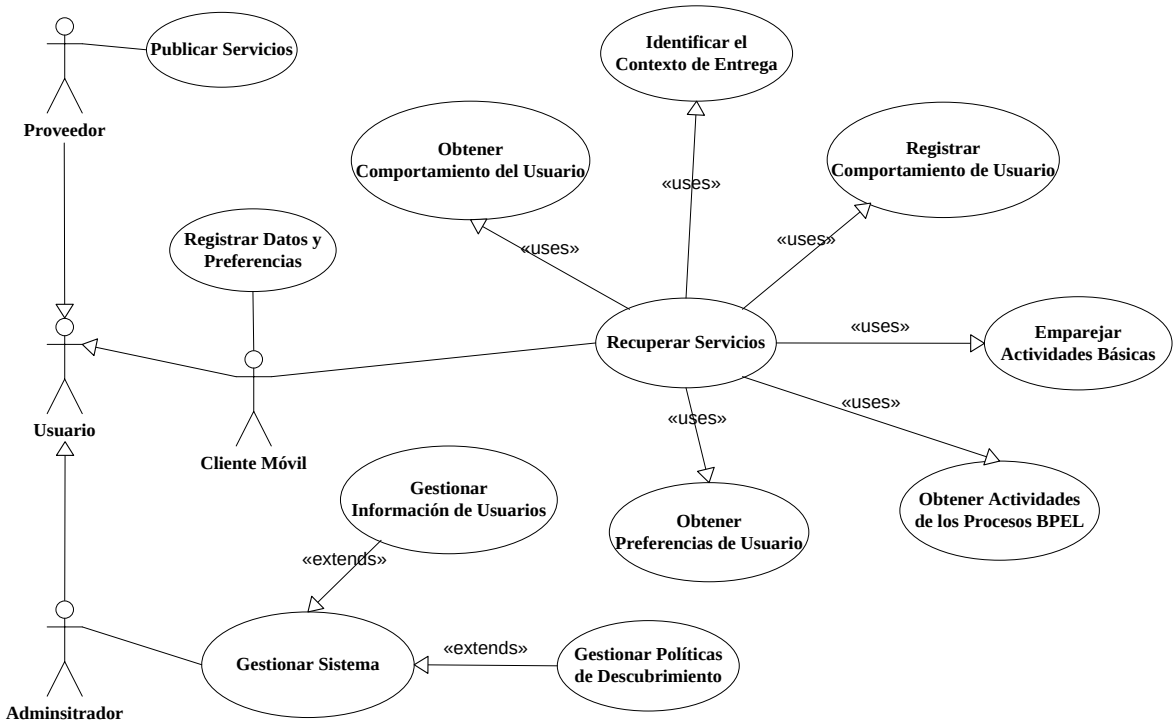


Figura 12. Diagrama de Casos de Uso del Sistema

4.2. Descripción de la solución

En esta fase se llevan a cabo aspectos relacionados con el desarrollo y la construcción de la plataforma final, tales como Modelado y diseño de los prototipos. Para ello se hace uso de UML.

4.2.1. Diagramas de clases del Sistema

En esta sección se expone los diagramas de clases (sin los atributos y métodos que lo conforman, con el objetivo de hacer legible su presentación en éste documento) de las aplicaciones software definidas que componen la plataforma U-ServiceMatch: *ServiceMatch*, *SeMatch-Contex* y *Repositorio de Usuarios.* Las descripciones de las clases de cada diagrama son también presentadas en esta sección.

4.2.1.1. Diagrama de Clases de la aplicación ServiceMatch

La Figura 13 presenta el diagrama de clases de la aplicación ServiceMatch, las cuales han sido agrupadas según la funcionalidad que implementen: Gestión del Repositorio de Servicios, Emparejamiento de Servicios, Manejo de Grafos y Presentación.

- **Gestión del Repositorio de Servicios:** a continuación se describen las clases que implementan las funcionalidades que permiten manipular el *Repositorio de Servicios:*

  - *BPELRepositoryManager*: contiene la lógica que permite gestionar, tanto las consultas como la colección de extensiones EMF (*Eclipse Modeling Framework*) del Repositorio de Procesos BPEL.

- *DefaultEMFExtensionCollection*: una de las más importantes características del Repositorio de Procesos de Negocio, es la posibilidad de registrar múltiples modelos EMF, para soportar el almacenamiento de diferentes tipos de archivos, extendiendo así su capacidad. Ésta clase permite registrar los modelos para BPEL, WSDL, XSD (por defecto soportados dentro del repositorio) y otros. Para el caso del meta-dato definido en el presente trabajo de grado, se adicionó una nueva función que permite registrar la nueva extensión (*.context*) del archivo que contiene las características de contexto.

- *KentOCLQueryEngineImpl*: implementa el motor de consulta principal de la versión pública del *Repositorio de Servicios,* desarrollado por la Universidad de Kent. Es un motor de consulta OCL de código abierto que implementa el estándar OCL 2.0. Para éste prototipo, el motor OCL de Kent obtiene las URI de los archivos BPEL, almacenados en el repositorio de procesos de negocio. Debido a las características limitadas de este motor, en cuanto a la consulta de los datos contenidos dentro de los nuevos archivos registrados en el *Repositorios de Servicios*, se implementó una nueva funcionalidad que permite obtener la información de contexto del archivo *features.context*, asociado al proceso BPEL.

- *BPELRepository*: permite inicializar el Repositorio de Procesos, registrando tanto objetos EMF como los motores OCL de consulta.

- *BpelReader*: carga los archivos *.bpel* encontrados en el repositorio, los lee, mediantes lectores especializados para cada actividad, y almacena toda la información necesaria para el proceso de transformación. Para cada parte del archivo de entrada, se extraen sus componentes y se crean objetos que representan finalmente el procesos de negocio.

- *ProcessGraph*: permite almacenar en un meta-modelo de objetos la estructura resultante de la serialización del documento BPEL, representando finalmente un grafo mediante arcos, funciones, conectores y nodos *start/end.* Las funciones son los nodos del grafo que representan acciones específicas. Los nodos *start/end* representan el nodo inicial y el nodo final del grafo. Los conectores son nodos que hacen posible las conexiones entre elementos al igual que pueden llevar a cabo diversas conexiones en el proceso.

- *BpelTransform*: implementa las estrategias de transformación de cada actividad BPEL, siendo la responsable de transformar el metadato de BPEL en un Objeto Java de tipo *ProcessGraph*.

- *ControlBPELTransform*: implementa la lógica de control que permite almacenar, encontrar, usar y definir archivos basados en el esquema estándar XML. Además, permite transformar archivos BPEL en su equivalente en Grafos.

- **Manejo de Grafos:** a continuación se describen las clases que implementan funcionalidades que permiten operar sobre los Grafos abstraídos de los procesos encontrados en el *Repositorio de Servicios*:

  - *ActivityRepository*: contiene la lógica para obtener las funciones/nodos de los Grafos de procesos, para organizarlos y almacenarlos por tipo de actividad (*invoke, reply, receive*).

  - *Activity*: representa la actividad/nodo/función de un Grafo. La cual puede ser: *FunctionInvoke*, *FunctionReceive*, *FunctionReply*. Cada función posee cuatro atributos: *Tipos de Actividad, Nombre de la operación PartnerLink* y *PortType.*

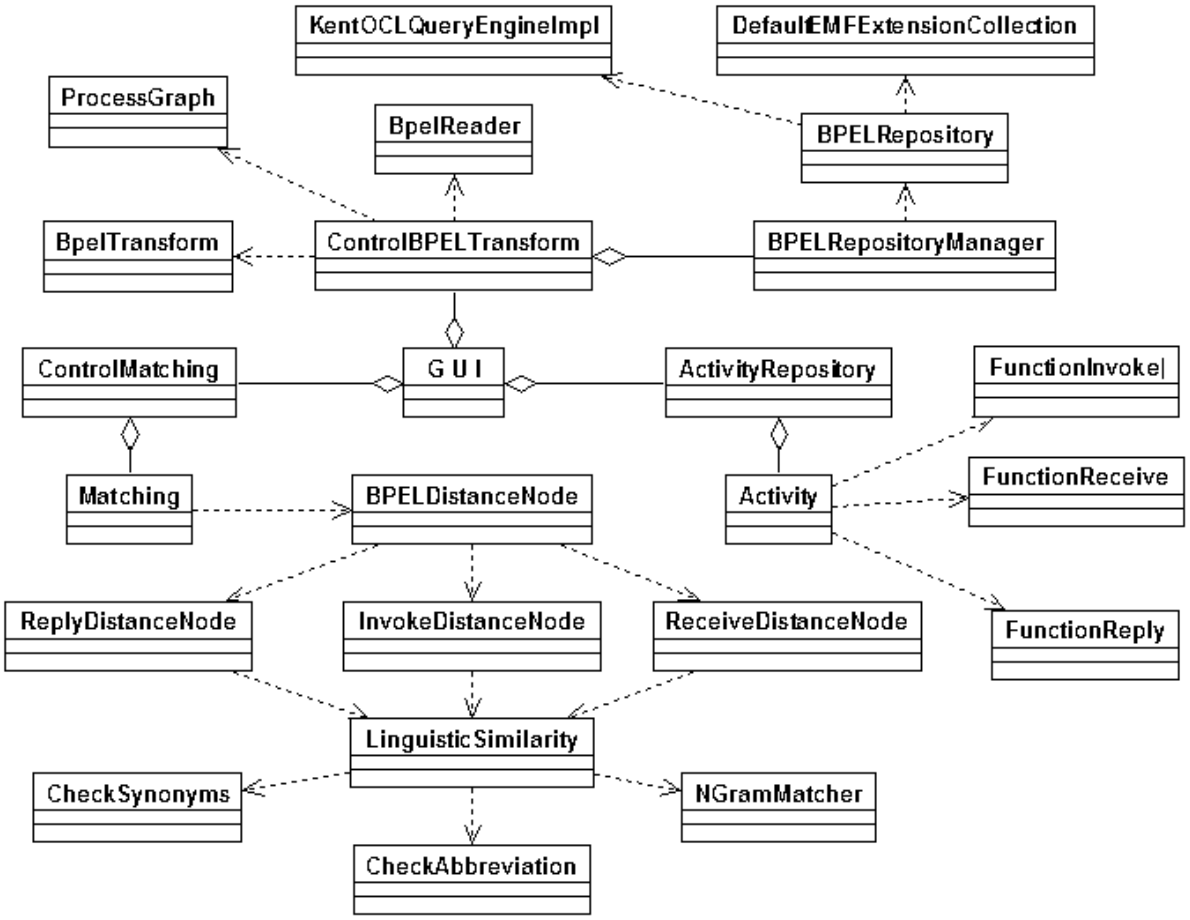


Figura 13. Diagrama de Clases aplicación ServiceMatch

- **Emparejamiento de Servicios:** a continuación, se describen las clases que implementan funcionalidades que soportan el proceso de emparejamiento de servicios:

  - *Matching*: toma como entradas dos nodos, que representan actividades básicas de BPEL (*receive, invoke, reply*), y calcula la distancia semántica entre ellos. Para esto, utiliza las clases: *InvokeDistanceNode*, *ReceiveDistanceNode y ReplyDistanceNode*; quienes a su vez usan la clase *LinguisticSimilarity.*

- *BPELDistanceNode*: implementa la lógica que permite verificar que los nodos, a ser comparados, tengan el mismo tipo de actividad.

- *InvokeDistanceNode*: permite obtener la distancia semántica entre dos nodos tipo *Invoke*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *ReceiveDistanceNode*: permite obtener la distancia semántica entre dos nodos tipo *Receive*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *ReplyDistanceNode*: permite obtener la distancia semántica entre dos nodos tipo *Reply*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *LinguisticSimilarity*: calcula la similitud lingüística entre dos etiquetas basándose en sus nombres. Para obtener esta medida se utilizan los algoritmos *Ngram, Check synonym* y *Check abbreviation*, implementados por las clases *CheckAbbreviation*, *CheckSynonyms* y *NGramMatcher* respectivamente.

- *CheckAbbreviation*: implementa la lógica que le permite estimar la similitud de dos cadenas, usando un diccionario de abreviaciones adecuado al dominio de aplicación.

- *CheckSynonyms*: implementa la lógica que le permite estimar la similitud de dos cadenas, usando el diccionario lingüístico WordNet, para identificar sinónimos.

- *NGramMatcher*: implementa la lógica que le permite estimar la similitud de dos cadenas, de acuerdo al número común de *qgramas* entre las etiquetas.

❖ **Presentación:** a continuación se describen las clases que implementan funcionalidades de representación visual del prototipo *ServiceMatch*:

- *GUI*: ésta clase implementa la lógica de presentación para las interfaces gráficas del prototipo *ServiceMatch*.

- *ControlMatching*: ésta clase implementa el control de los datos entre los procesos de tratamiento y visualización de la información.

4.2.1.2. Diagrama de Clases de la aplicación SeMatch-Context

La Figura 14, presenta el diagrama de clases de la aplicación *SeMatch-Context*, las cuales son descritas a continuación:

- *ContextQueryEngine*: ésta clase es utilizada por *ControlBPELTransform* de la aplicación *ServiceMatch* expuesta, la cual implementa un mecanismo de consulta del valor

correspondiente al atributo de contexto (*Tipo de Acceso*), de los servicios publicados en el repositorio.

- *ServiceMatch*: representa el *.JAR* (*Java ARchive*) del prototipo ServiceMatch definido. Los *.jar* son un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Dada la lógica implementada en el *ServiceMatch*, permite determinar la similitud entre un nodo *Query* y una colección de nodos *Target* (1:N), y entregar una lista ordenada de las actividades más similares a un nodo de consulta.

- *ServiceDiscovery*: su función es atender las peticiones provenientes de un cliente móvil a través de su navegador Web. Esta clase implementa la lógica que le permite al prototipo *SeMatch-Context,* obtener el contexto de entrega de los clientes que acceden a la aplicación. Además, controla el proceso de recuperación de servicios, considerando tanto el requerimiento del contexto de los servicios, como las restricciones de contexto del usuario.

- *DeliveryContext*: se encarga de obtener el contexto de entrega, utilizando tres fuentes de información: Cabeceras HTTP, UAProf y WURFL.

- *GeneralCapabilityExplorer*: permite explorar las capacidades de un dispositivo móvil, obteniendo la información de la biblioteca WURFL o de los repositorios UAProf.

- *UAProfCapabilityExplorer*: se encarga de explorar las capacidades de un móvil obteniendo la información en los repositorios UAProf.

- *WurflCapabilityExplorer*: se encarga de explorar las capacidades de un móvil, obteniendo la información en la biblioteca WURFL.

- *CapabilityExplorer*: es la Interfaz que define la estructura de los exploradores de capacidades para dispositivos móviles.

- *Init*: carga en RAM el archivo "*wurfl.xml*", que contiene la información de la biblioteca WURFL.

- *Control*: controla la obtención de información en las bibliotecas WURFL.

- *FilterWurfl*: ésta clase permite buscar las capacidades de un dispositivo en la biblioteca WURFL mediante la cabecera HTTP, "*User-Agent*".

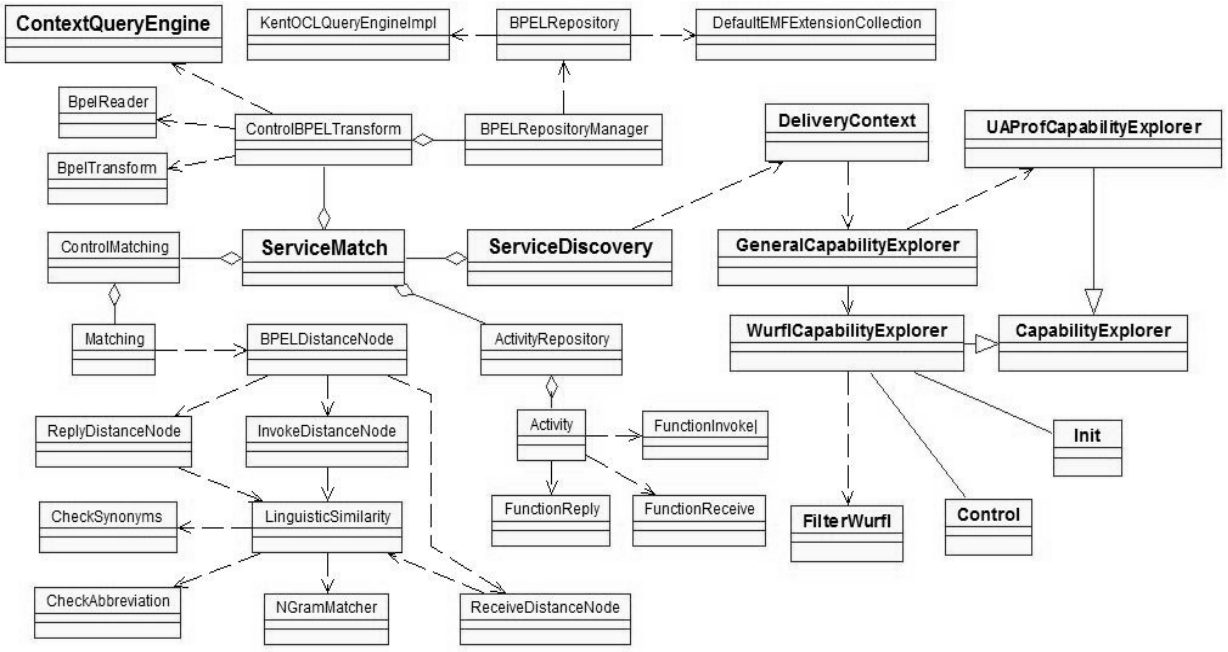


Figura 14. Diagrama de Clases aplicación SeMatch-Context

4.2.1.3. Diagrama de Clases del Repositorio de Usuarios

En la Figura 15, se observa las clases que conforman al repositorio de Usuarios, las cuales son descritas a continuación:

- *UserRepository*: encargada de Gestionar y exponer las funcionalidades que ofrece el Repositorio de usuario, especialmente la de sugerencia de servicios, basado en la información de usuario.

- *UserRegistryGUI*: es la interfaz por medio de la cual el usuario puede registrar su información, en éste momento es una interfaz Web.

- *Register*: brinda el manejo de la información del usuario, registra su información de perfil, como también los servicios que consume a medida que utiliza la plataforma.

- *UserProfile:* representa la información del usuario, según las especificaciones de (Guerrero, et al., 2010).Con base en la similitud de los parámetros de esta clase, se encuentra los perfiles más similares al del usuario.

- *UserProfilePersistence*: encargada de controlar la persistencia de *UserProfile*

- *QueryManager*: encargada de manejar la información de las consultas realizadas previamente por los usuarios.

- *Query*: representa una consulta ya realizada, tiene como componentes la lista de nodos que fueron sugeridos para esa oportunidad.

- *QueryPersistence*: encargada de controlar la persistencia de la clase *Query*.

- *ControlMatchingP:* gestiona el proceso de descubrimiento de los perfiles más similares, dentro de los almacenados en el repositorio.

- *MatchingP*: es la encargada de implementar el matching atómico de perfiles.

- *PoliticyManager*: gestiona las variables del sistema con relación a los pesos de los parámetro,s para el emparejamiento tanto de Actividades como de Usuarios.

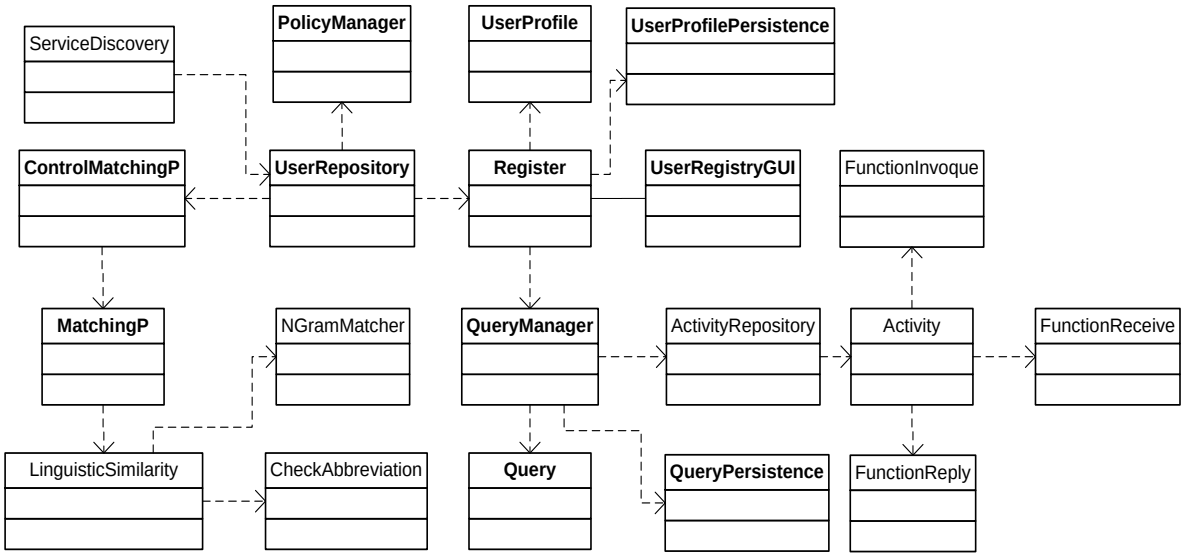


Figura 15. Diagrama de Clases del Repositorio de Usuarios

4.2.2. Diagramas de paquetes del Sistema

Los diagramas presentados en esta sección, exponen la vista lógica de las aplicaciones software que componen el sistema. Dichos diagramas se organizan en paquetes, subsistemas y capas (*Aplicación, mediación, y Almacenamiento y Servicios*) (Jacobson, et al., 1998). Además, se muestra la interacción existente entre capas, así como los paquetes más relevantes que las componen.

4.2.2.1. Diagrama de Paquetes para la aplicación ServiceMatch

La Figura 16, presenta la vista lógica de la aplicación *ServiceMatch*. A continuación se describen las capas y su interacción, así como los paquetes más relevantes que las componen.

- **Capa de aplicación:** contiene los paquetes que implementas las funcionalidades del prototipo, estos son:

  - *Analizador BPEL-Grafos*: permite registrar funciones que transforman meta-modelos de servicios en su equivalente en grafos. Para el prototipo *ServiceMatch* se ha

implementado las funciones de BPEL. Sin embargo, es posible registrar otro tipo de funciones y extender su aplicabilidad a otros modelos como WSDL o WS-CDL.

- *Clasificador de Actividades*: contiene las clases que implementan la lógica para obtener y clasificar los nodos, tanto del grafo de entrada, como de los abstraídos del *Repositorio de Servicios*. Para ellos usa el *Almacenador de Nodos por tipo de Actividad* y el *Filtro de Actividades*.

- *Almacenador de Nodos por tipo de Actividad*: contiene las clases que permiten obtener y clasificar los nodos de un grafo por tipo de actividad (*receive, invoke* y *reply*). Paralelamente, almacena los nodos obtenidos, verificando, con ayuda del *Filtro de Actividades,* que no estén repetidos.

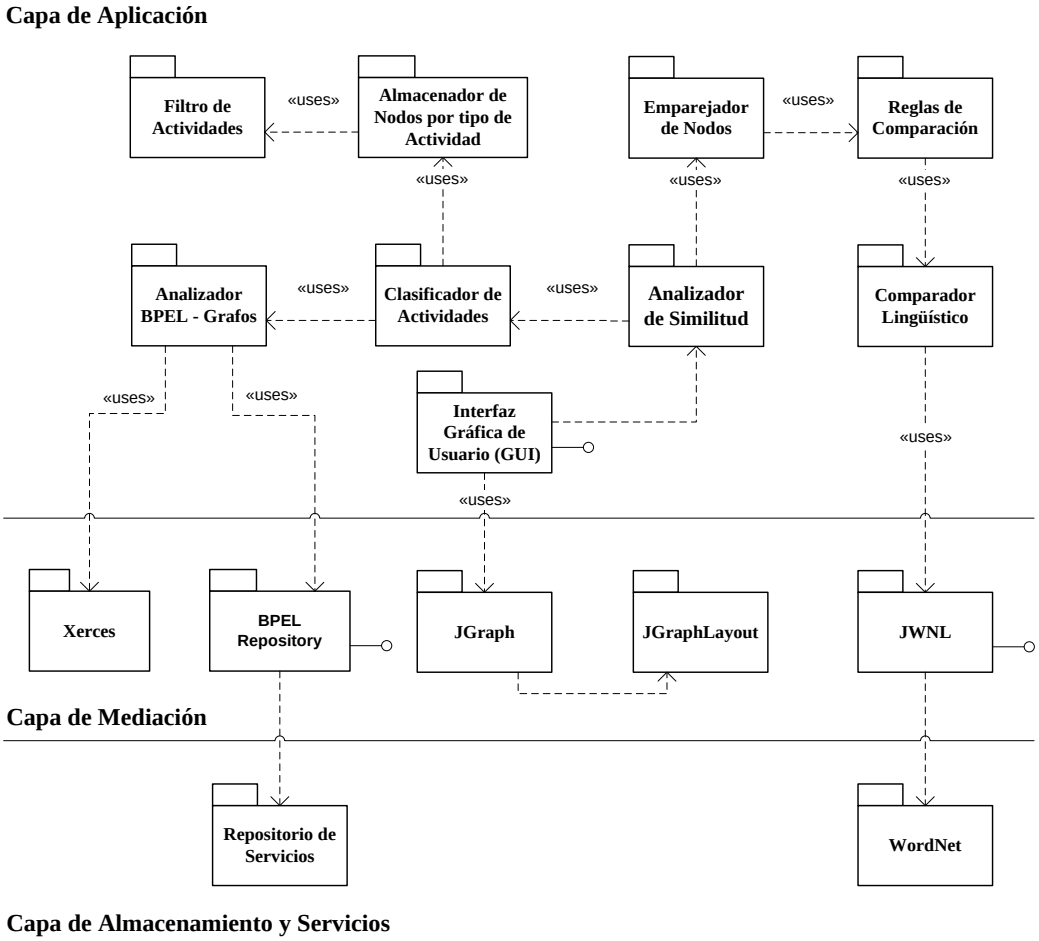


Figura 16. Vista lógica de la Aplicación ServiceMatch

- *Filtro de Actividades:* contiene las clases que implementan un algoritmo que filtra las actividades que ya han sido almacenadas.

- *Analizador de Similitud*: contiene las clases que implementan la lógica para determinar la similitud entre un nodo *Query* y una colección de nodos *Target* (1:N), y

así entregar una lista ordenada de las actividades más similares al nodo de consulta. Éste paquete usa el *Emparejador de Nodos, Reglas de Comparación* y el *Comparador Lingüístico* para calcular dicha coincidencia.

- *Emparejador de Nodos*: implementa la lógica que permite determinar la similitud de dos nodos *Query* y *Target*, basado en la distancia semántica (proporcionada por los módulos *Reglas de Comparación* y *Comparador Lingüístico*) entre ellos.

- *Comparador Lingüístico*: contiene las clases que implementan la lógica para calcular la similitud entre dos cadenas. Por ejemplo, para el *ServiceMatch* se ha utilizado los algoritmos *Ngram*, *Sinonym* y *Abbreviation*. Sin embargo, otros algoritmos para tal fin pueden ser registrados.

- *Interfaz Gráfica de Usuario (GUI)*: Con el fin de lograr una representación visual, éste paquete contiene todas las clases que implementan las interfaces gráficas del prototipo *ServiceMatch*.

❖ **Capa de Mediación:** contiene todas las interfaces de programación de aplicaciones (API) utilizadas por el prototipo. La capa está compuesta por los siguientes paquetes:

- *Xerces*: es un Parser XML compuesto por una familia de paquetes software, que permiten analizar y manipular archivos XML. La biblioteca implementa una serie de API estándar que incluye a DOM[5], SAX[6] y SAX2.

- *BPEL Repository*: es una API de Java que permite manipular documentos BPEL (y otros archivos XML relacionados), encontrados en el *Repositorio de Servicios,* como objetos. Ocultando la serialización y deserialización al usuario. Las consultas son realizadas utilizando un lenguaje de consulta orientado a objetos, OCL[7].

- *JGraph:* es una librería que contiene todas las funcionalidades de visualización e interacción gráfica. Éste paquete permite la visualización de la interfaz gráfica de usuario (GUI).

- *JgraphLayout*: es una librería de diseño gráfico de alto rendimiento para Jgraph, que automáticamente posiciona una gráfica, un diagrama, o una red de una manera visualmente agradable.

- *JWNL* (Java WordNet Library:) es una API que permite acceder al diccionario relacional WordNet. También provee funcionalidades más allá del acceso a datos, tales como el descubrimiento de relaciones y procesamiento morfológico.

❖ **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento del prototipo. Esta capa se compone de los siguientes paquetes:

- *Repositorio de Servicios*: Es un repositorio de procesos de negocio, que soporta la recuperación de archivos BPEL (y otros documentos XML) y provee un poderoso

[5]DOM: Document Object Model. Es un Modelo en Objetos para la representación de Documentos HTML y XML.
[6]SAX: Simple API for XML. Provee mecanismos que para leer datos de um documento XML.
[7]OCL: Object Constraint Language.

mecanismo de consulta que permite encontrar un proceso con búsquedas en sus propiedades o metadatos relacionados. Este repositorio provee una API (paquete *BPEL Repository*) con el fin de hacer flexible su uso.

- *WordNet*: es una enorme base de datos léxica del idioma inglés. Agrupa las palabras en conjuntos de sinónimos llamados 'synsets', proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos. El objetivo es de éste proyecto es: por un lado producir una combinación de diccionario y tesauro cuyo uso es más intuitivo, y ayudar al análisis automático de textos y a las aplicaciones de inteligencia artificial. Este paquete es utilizado por el *Comparador Lingüístico* (a través del paquete JWNL) para verificar la relación semántica entre dos palabras.

4.2.2.2. Diagrama de Paquetes de la aplicación SeMatch-Context

La Figura 17, presenta la vista lógica de la aplicación SeMatch-Context.A continuación se describen lascapas y su interacción, así como los paquetes más relevantes que las componen.

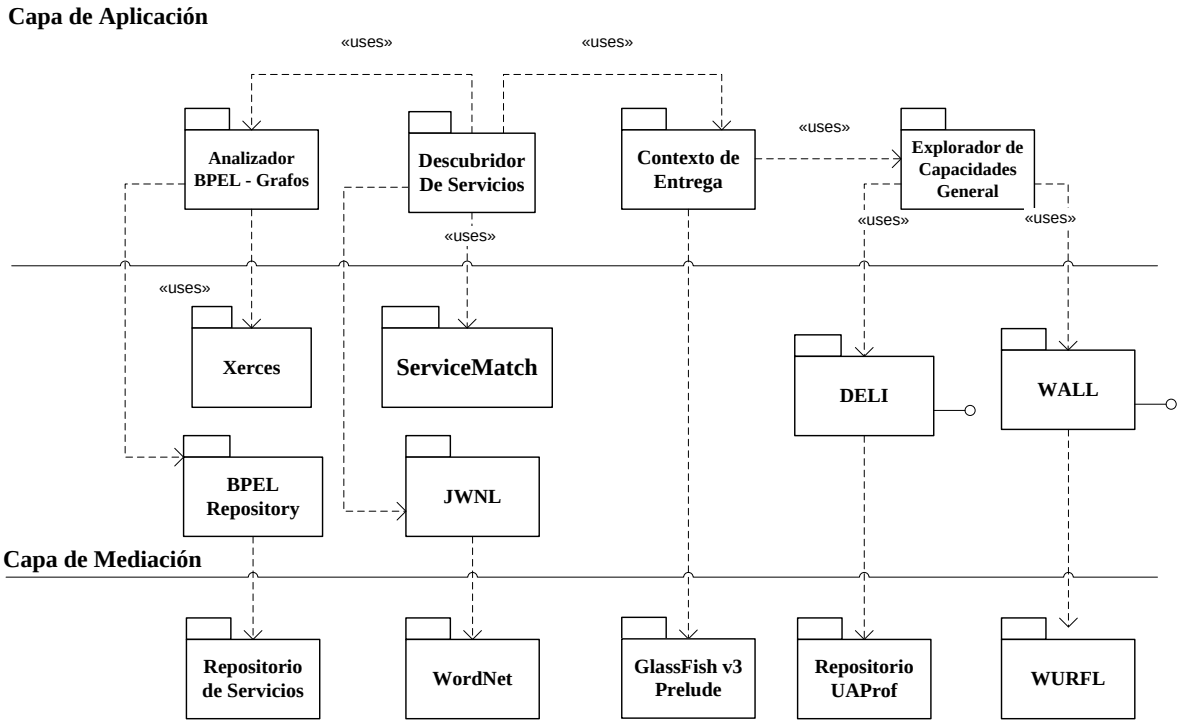


Figura 17. Vista lógica de la Aplicación SeMatch-Context

- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades de la aplicación. Ésta capa está compuesta por:

- *Contexto de Entrega*: permite obtener el contexto de entrega de los clientes móviles que acceden a la aplicación, tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. A partir de esta información, realiza un proceso de descarte entre actividades aptas para el consumo y las que no lo son.

- *Explorador de Capacidades General*: permite explorar las capacidades de un dispositivo móvil, obteniendo la información de la biblioteca WURFL o de los repositorios UAProf.

- *Service Discovery:* contiene la clase *ServiceDiscovery*, encargada de implementar la lógica que permite recuperar los servicios más relevantes para la consulta de un usuario, mediante una fase emparejamiento a nivel atómico de las actividades BPEL, contenidas en el *Repositorio de Servicios*. Luego, utilizando el módulo de Contexto de Entrega, ejecuta el proceso de descarte y así obtiene las actividades consumibles por el dispositivo solicitante.

❖ **Capa de Mediación:** contiene las interfaces de programación de aplicaciones (APIs) utilizadas por la aplicación.

- *DELI*: librería que permite a un Servlet Java conocer el contexto de entrega de un dispositivo a través de CCPP o UAProf. El contexto de entrega ayuda a los servidores a identificar en qué clase de entorno se está realizando la petición y con base en esto puede dar una respuesta más adecuada según el dispositivo.

- *WALL*: *Wireless Abstraction Library* (Librería de Abstracción Móvil) es una librería de tags JSP que permite diseñar páginas Web que, dependiendo de las capacidades del dispositivo, entrega el contenido en formato WML[8], C-HTML[9], y XHTML[10] Mobile Profile.

- *ServiceMatch*: representa el *.JAR* (*Java ARchive*) del prototipo *ServiceMatch* definido. Los *.jar* son un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Dada la lógica implementada en el *ServiceMatch*, permite determinar la similitud entre un nodo *Query* y una colección de nodos *Target* (1:N), y entregar una lista ordenada de las actividades más similares a un nodo de consulta.

❖ **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento de la aplicación. Esta capa se compone de los siguientes paquetes:

- *GlassFish v3Prelude*: es un servidor de aplicaciones open source, desarrollado por Sun Microsystems, que implementa las tecnologías definidas en la plataforma Java EE 6 y permite ejecutar aplicaciones que siguen esta especificación.

- *WURFL*: *Wireless Universal Resource FiLe* (Archivo de Recursos Universal Inalámbrico). Es parte del esfuerzo de una comunidad FOSS (Free and Open Source Software, Código Fuente Libre y Abierto), enfocada en el problema de presentar

[8] WML: Wireless Markup Language (Lenguaje de marcado inalámbrico)
[9] C-HTML: Compact HyperText Markup Language (Lenguaje de marcado de hipertexto compacto)
[10] XHTML: eXtensible HyperText Markup Language (Lenguaje de marcado de hipertexto extensible)

contenido en la amplia variedad de dispositivos móviles. El WURFL es un fichero de configuración XML, que contiene información acerca de características y capacidades para una variedad de dispositivos móviles. Dicha información es una contribución de desarrolladores de todo el mundo y el WURFL es actualizado de forma frecuente, reflejando los nuevos dispositivos móviles que entran en el mercado

- *Repositorio UAProf*:es una especificación basada en el CCPP, que se encarga de describir las preferencias de usuario y las capacidades de los dispositivos, con el fin de obtener una experiencia de usuario, más personalizada.

4.2.2.3. Diagrama de Paquetes de U-ServiceMatch

En la Figura 18 se presenta la vista lógica de toda la plataforma U-ServiceMatch, especialmente la parte de personalización, para mostrar su integración con los demás componentes.

- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades de la aplicación. Ésta capa está compuesta por:

  - *Service Discovery:* contiene la clase ServiceDiscovery, encargada de implementar la lógica que permite recuperar los servicios más relevantes, para la consulta de un usuario. Ésta clase, en la última fase de integración, hace uso del módulo de Perfil de Usuario para conseguir un número menor de servicios sugeridos, para ser organizados mediante una fase emparejamiento a nivel atómico. Si estos servicios no tuvieran las características necesitadas, se realiza una búsqueda sobre las actividades BPEL contenidas en el *Repositorio de Servicios.* luego utilizando el módulo de Contexto de Entrega se ejecuta el proceso de descarte y así se obtienen las actividades consumibles por el dispositivo solicitante.

  - *User Repository:* representa el *.JAR*,que contiene las clases relacionadas con la lógica del proceso de sugerencia de servicios, éste componente brinda todas las funciones que el módulo de descubrimiento necesita, para soportar la personalización de su proceso. También soporta las operaciones que UserGUI realice sobre la información de usuario.

  - *UserGUI:* ofrece una interfaz al cliente, ésta interfaz le permite registrar sus datos personales (perfil de usuario)

- **Capa de Mediación:** contiene las interfaces de programación de aplicaciones (APIs) utilizadas por la aplicación.

  - *UserRepositoryPersistence*: es el componente encargado de brindar el soporte a la persistencia de los datos de usuario.

- **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento de la aplicación. Esta capa se compone de los siguientes paquetes:

  - PostgreSQL: es un sistema de gestión de bases de datos objeto-relacional (ORDBMS), utilizado para generar las tablas, que contienen la información relacionada con el usuario.

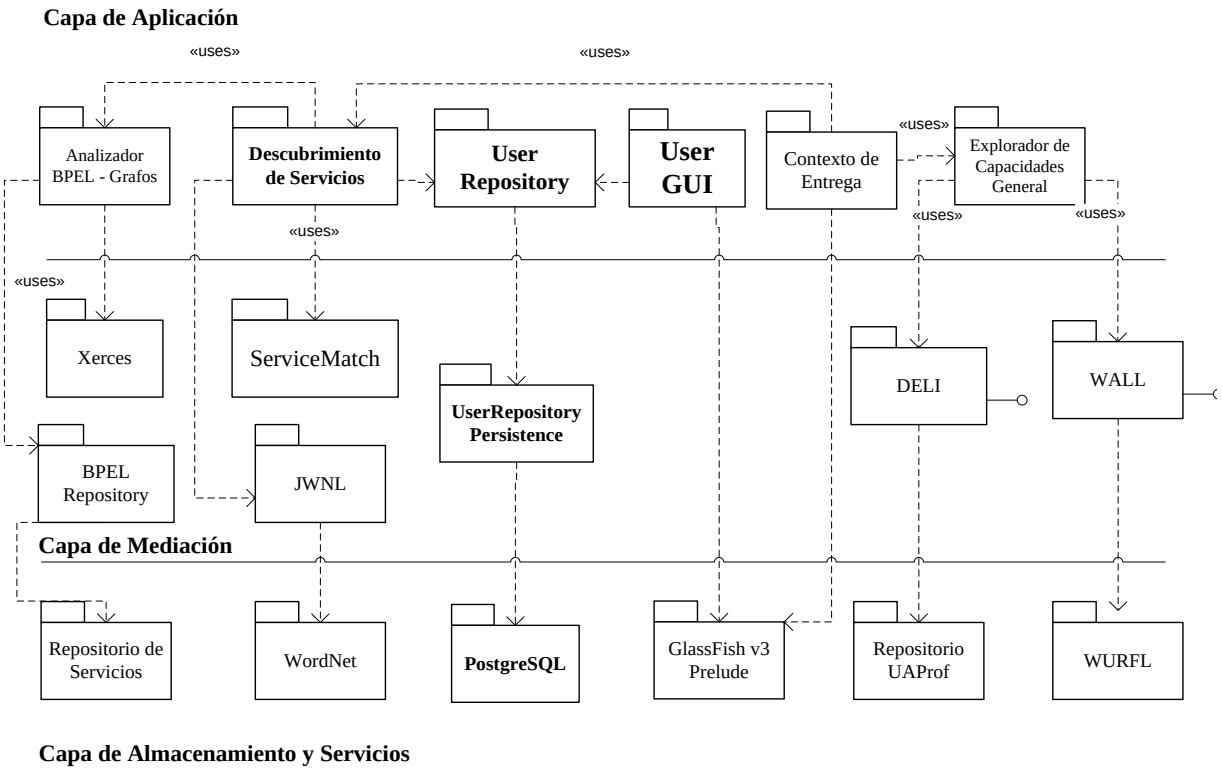


Figura 18. Diagrama de Paquetes U-ServiceMatch

4.3. Definición de la Arquitectura

En la Figura 19, se presenta la arquitectura que da soporte a la plataforma U-ServiceMatch. A continuación, se da una breve descripción de cada uno de los subsistemas que la componen.

Solicitante: corresponde a los clientes móviles que interactúan con la plataforma por medio su navegador Web (Browser), con el fin de consultar, invocar y consumir los servicios de la red ubicua.

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, tales como: entradas, salidas, precondiciones, restricciones, etc. éste módulo es implementado a través de una aplicación web.

Parser BPEL a Grafos: permite transformar las descripciones de comportamiento (BPEL) en su equivalente en grafos. 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 (**Anexo A.2.1.1**).

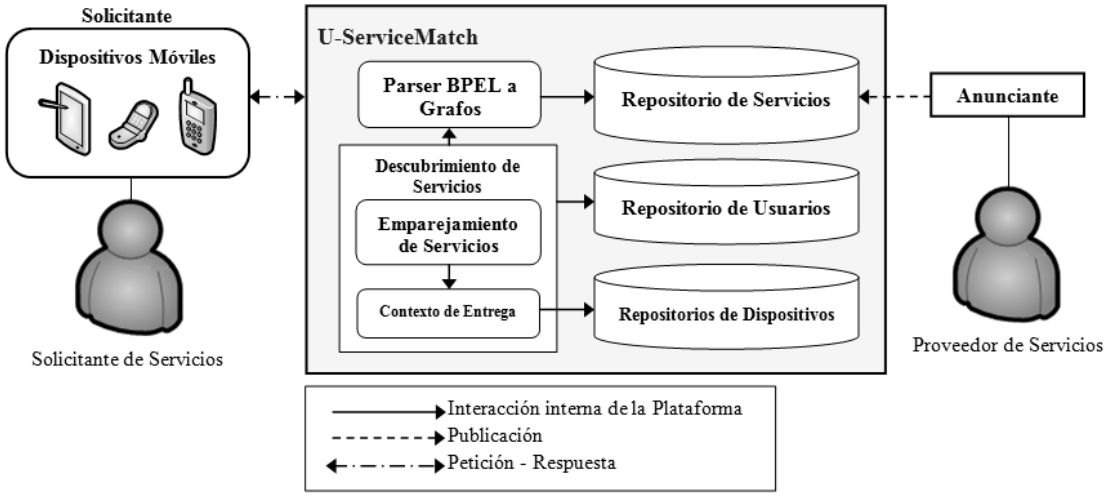


Figura 19. Arquitectura de la Plataforma U-ServiceMatch

Módulo de Descubrimiento de Servicios: recupera los servicios más apropiados del *Repositorio de Servicios,* para posteriormente componer el servicios requerido por el usuario, basado en un algoritmo de emparejamiento de actividades básicas BPEL (*Módulo de Emparejamiento de* Servicios) y considerando tanto las preferencias del usuario (*Repositorio de Usuarios*), como las restricciones del contexto del solicitante (*Módulo de Contexto de Entrega*).

Módulo de Emparejamiento de Servicios: apoyado en el *Parser BPEL a Grafos*, éste módulo emplea una representación formal de grafos, para los requerimientos de usuarios y los servicios almacenados en el repositorio, tornando el problema de emparejamiento de actividades básicas BPEL, a 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 el *Repositorio de Servicios.* Toda la funcionalidad e implementación de este módulo es expuesta en **Anexo A.3.1**.

Módulo de Contexto de Entrega: actúa una vez recuperados los servicios más relevantes, verificando si las actividades recuperadas pueden ser invocadas en el contexto del usuario (**Anexo A.3.2.1**).

Repositorio de Servicios: provee un poderoso mecanismo de publicación(**Anexo B.1**) y consulta (**Anexo A.3.2.2**)de documentos BPEL y otros documentos XML, los cuales represan respectivamente los servicios de la red ubicua y las descripciones de requerimiento de contexto.

Repositorio de Dispositivos: provee un mecanismo que identifica las características de los dispositivos que acceden al sistema, tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. Esta información es recuperada de los repositorios UAProf (*User Agent Profile*) y la librería WURFL (*Wireless Universal Resource*) (**Anexos A.3.2 y A.2**).

Repositorio de Usuarios: almacena información relacionada con los usuarios del sistema, como preferencias explicitas provistas por los usuarios o datos implícitos capturados

dinámicamente tales como, historial de servicios consumidos o patrones de uso (**ANEXO A.2.1.1**).

4.3.1. Diagrama de Despliegue de Plataforma U-ServiceMatch

El diagrama de despliegue expuesto en la Figura 20, describe la configuración de la Plataforma U-ServiceMatch para su ejecución en un ambiente del mundo real, presentando la disposición física de los distintos nodos que intervienen en la composición del Sistema y la distribución de los programas ejecutables sobre ellos. Esto permite modelar aspectos físicos, como la vista estática de despliegue del sistema, la configuración de nodos y los componentes que residen en ellos, y la topología hardware donde se ejecuta la aplicación.

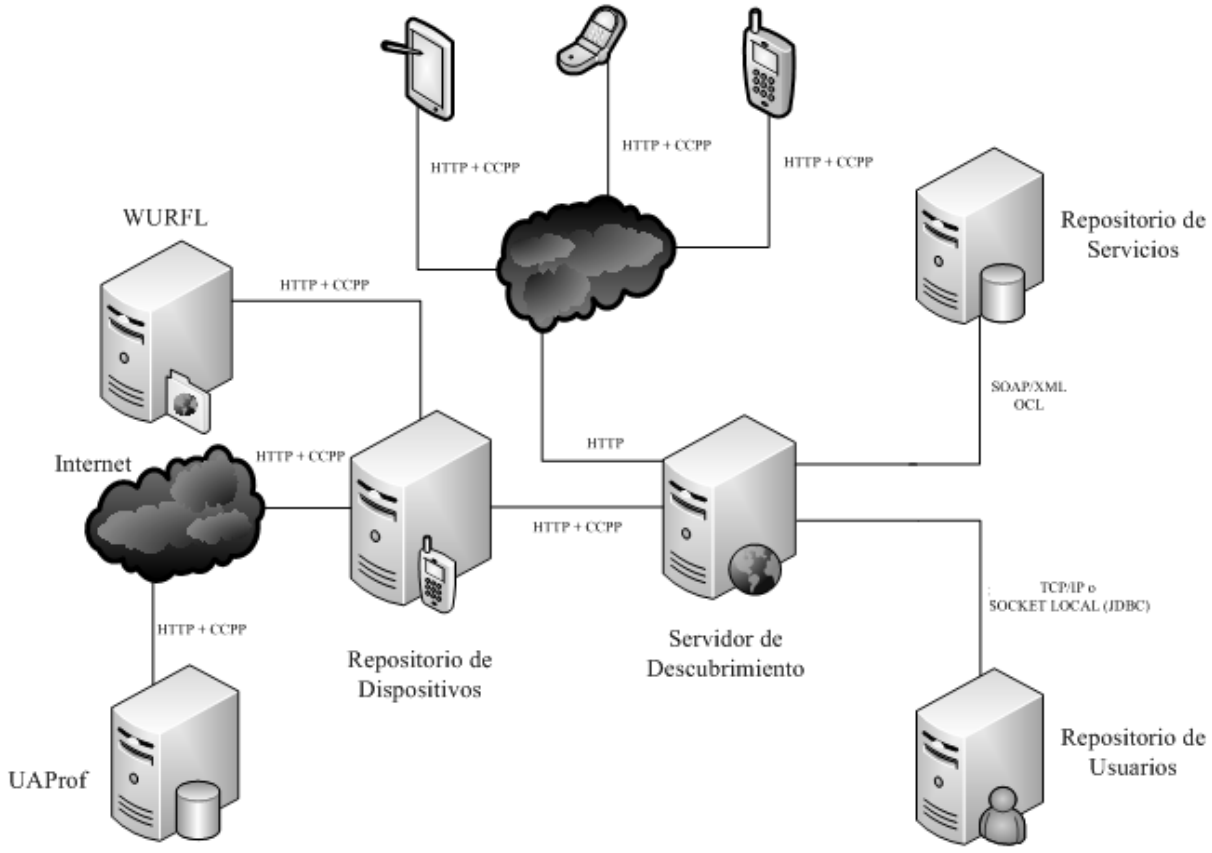


Figura 20. Diagrama de Despliegue de la Plataforma U-ServiceMatch

Como se puede observar el sistema se encuentra totalmente distribuido, su funcionamiento se describe a continuación.

Los clientes móviles acceden a la plataforma mediante el navegador Web, enviando una URL por medio de las cabeceras HTTP, la cual contiene capacidades básicas del dispositivo descritas según el estándar CC/PP. Ésta petición es transmitida por la *World Wide Web* hasta el *Servidor de Descubrimiento,* que identifica las capacidades básicas del dispositivo, que está interactuando con la plataforma por medio de las cabeceras HTTP. Sin embargo, dicha información no es suficiente, es aquí donde interviene el *Repositorio de Dispositivos*, el cual determina el contexto de entrega completo del cliente móvil.

El contexto de entrega es construido con base en tres fuentes de información UAProf, WURFL y las cabeceras HTTP, lo que garantiza una descripción amplia y fiel de las capacidades del dispositivo. El orden en el cual el *Repositorio de Dispositivos* obtiene el contexto de entrega es el siguiente: El Servidor inicialmente usa la información que envía el *Servidor de Descubrimiento* por medio de las cabeceras HTTP, luego los datos ofrecidos por la librería WURFL y por último usa UAProf. Se ha considerado que éste es el orden de fidelidad de la información, ya que el navegador debe conocer fielmente sus capacidades, y la librería WURFL ha sido construida por desarrolladores que mediante pruebas obtienen éstos datos; mientras que en el caso de UAProf suele obtenerse información que no detalla el producto, sino una categoría de dispositivos del fabricante.

Una vez el contexto de entrega del cliente móvil ha sido identificado, el usuario es habilitado para usar la plataforma, permitiéndole consultar su historial de servicios (consumidos y consultados) o ejecutar una búsqueda para recuperar nuevos servicios. Esto es posible, gracias a los Repositorios de Usuarios y de Servicios del sistema, donde el primero gestiona los *Perfiles de Usuario*, registrando el comportamiento del mismo, servicios consumidos o historial y patrones de uso; y el segundo provee un poderoso mecanismo de consulta, que permite obtener información de los procesos BPEL almacenados, mediante búsquedas en sus propiedades o metadatos relacionados.

Protocolos de comunicación y conexión.

Los protocolos de comunicación, son el conjunto de reglas que especifican el intercambio de mensajes durante la comunicación entre las entidades que forman parte de una red. Los utilizados por la plataforma U-ServiceMatch son los siguientes:

- HTTP (*HyperText Transfer Protocol*): el Protocolo de transferencia de hipertexto define la sintaxis y la semántica, que utilizan los elementos software de la arquitectura web (clientes, servidores) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

- SOAP (*Simple Object Access Protocol*): el Protocolo Simple de Acceso a Objetos define cómo dos objetos en diferentes procesos pueden comunicarse por medio del intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

- XML (*eXtensible Markup Language*): el lenguaje de marcas extensible juega un papel fundamental en el intercambio de una gran variedad de datos. Su función principal es describir datos y permite la lectura de datos a través de diferentes aplicaciones. XML sirve para estructurar, almacenar e intercambiar información.

- OCL (*Object Constraint Language*):se utiliza como un lenguaje de consultas orientado a objetos, para los datos del *Repositorio de Servicios*. El poder de un enfoque orientado a objetos, es que la representación del objeto de datos sólo deben ser conocida cuando las consultas son formuladas. Los programas usan esta representación internamente y así, es sencillo construir consultas para esta representación de datos. No es necesario conocer cómo se almacenan los datos, por ejemplo, en una base de datos relacional, o en una base de datos XML.

- JDBC (*Java Database Connectivity*): es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede.

- TCP/IP (*Transmission Control Protocol/Internet Protocol*): protocolo de Control de Transmisión/Protocolo de Internet, el cual es un sistema de protocolos que hacen posibles servicios Telnet, FTP, E-mail, y otros entre ordenadores que no pertenecen a la misma red.

4.4. Resumen

En este capítulo, se expone el proceso de ingeniería realizado para definir la arquitectura que soporta la plataforma de descubrimiento de servicios en ambientes ubicuos, U-*ServiceMatch*. Obteniendo de las iteraciones planteadas, los Diagramas de Casos de Uso, Diagramas de Clases y Diagramas de Paquetes de las aplicaciones software que constituyen el Sistema.

Entre los aspectos teóricos expuestos en esté capitulo se describieron las consideraciones previas a la definición de la arquitectura y desarrollo del sistema-solución. Finalmente se presentó el diagrama de despliegue de U-ServiceMatch que describe la configuración de la Plataforma para su ejecución en un entorno real.

Capítulo V

EXPERIMENTACIÓN Y EVALUACIÓN

En éste capítulo, inicialmente se describe Be4Sed, herramienta de evaluación desarrollada para valorar la calidad del proceso de emparejamiento, en la plataforma definida en el presente trabajo de grado: U-ServiceMatch. Además, se expone la metodología de evaluación usada y las pruebas ejecutadas a la plataforma, para establecer la calidad de sus resultados y su rendimiento. Finalmente el análisis de los resultados, junto con las gráficas asociadas y los trabajos futuros son presentados.

5.1. Metodología de Evaluación

Para la valoración de la plataforma U-ServiceMatch se desarrolló una herramienta de evaluación de técnicas de recuperación de servicios, denominada Be4SeD (Benchmarking for Service Discovery), la cual implementa una metodología de *benchmarking,* para evaluar la calidad de recuperación de las técnicas de emparejamiento de servicios, tanto en entornos Web como en ambientes de computación ubicua. Antes de exponer y detallar las funcionalidades de Be4SeD, es conveniente aclarar los conceptos más relevantes entorno a la metodología empleada para la evaluación de las aplicaciones software, desarrolladas en el presente trabajo de grado.

5.1.1. Benchmarking

Es un proceso sistemático y continuo para evaluar los productos, servicios y procesos de trabajo de las organizaciones que son reconocidas como representantes de las mejores prácticas, con el propósito de realizar mejoras organizacionales(Spendolini, 1994).

Básicamente un *benchmarking* es el proceso encargado de comparar *benchmarks*. Un *benchmark* es el resultado de una evaluación hecha a un servicio, producto o proceso. Así, *benchmarking* es el proceso en el cual se puede comparar los resultados de evaluaciones hechas a distintos servicios, productos o procesos.

En informática un *benchmark,* es entendido como el resultado de la ejecución de un programa o un conjunto de programas en una máquina, que brindan información sobre el rendimiento de un elemento concreto o la totalidad del sistema.

A continuación se explica brevemente, la adaptación de los conceptos expuestos anteriormente a la funcionalidad de la herramienta Be4SeD, los cuales son ampliamente detalladas en las siguientes secciones.

La plataforma Be4SeD es una aplicación disponible en internet, que implementa un método que permite a los usuarios comparar manualmente los modelos de un servicio *Query* y un servicio *Target*, y posteriormente, crear un ranking de servicios de acuerdo a los resultados de la comparación, a esto le denominamos *Benchmark de Referencia*. Estos modelos representan las características más relevantes de los servicios descritos en BPEL (proceso abstracto). De esta manera, la herramienta permite comparar el resultado obtenido por la

plataforma de descubrimiento de servicios (*Benchmark del Algoritmo*), con el ranking definido por los usuarios (*Benchmark de Referencia*). Por lo tanto, la herramienta Be4SeD está compuesta de un mecanismo de evaluación intuitivo, de un módulo de ingreso de los datos correspondientes al algoritmo a evaluar y un componente que entrega resultados estadísticos como: *recall, precision, overall, k-precision y p-precision*, medidas que permiten determinar la calidad de la técnica evaluada. Además, las funcionalidades de Be4SeD, se ofrecen como servicio web para facilitar la integración con las implementaciones de los algoritmos a evaluar.

5.2. Plataforma de Evaluación Be4SeD

La Figura 21 presenta los subsistemas de Be4SeD. En esta herramienta, evaluadores expertos en el tema de descubrimiento de servicios, comparan manualmente por parejas los servicios contenidos en el repositorio, con el fin de generar su propio *benchmark*. Una vez todos los evaluadores registrados en la plataforma concluyen la evaluación de los servicios, el administrador de Be4SeD ordena la creación del *Benchmark de Referencia*, ejecutando las políticas que permiten generalizar los resultados de la evaluación de cada experto. Éste *Benchmark de Referencia* es la característica más relevante de Be4SeD, ya que para evaluar y determinar la calidad de un algoritmo, es necesaria una base confiable que pueda ser comparada con los resultados arrojados por la técnica de emparejamiento y así inferir sobre la calidad de los mismos.

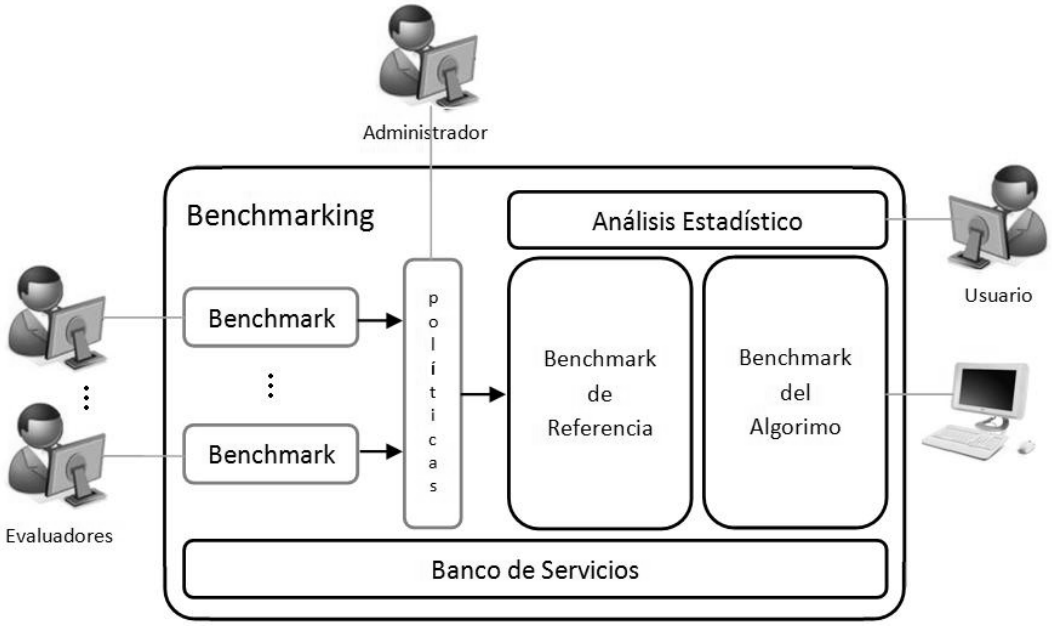


Figura 21. Arquitectura genérica de Be4SeD

Por otro lado, Be4SeD permite a los autores de las diferentes técnicas de recuperación de servicios (usuarios) crear su propio *Benchmark del Algoritmo,* con el fin de comparar los resultados con el *Benchmark de Referencia* generado por los expertos evaluadores. Por último, los usuarios también pueden acceder al sistema de *Análisis Estadístico*, para obtener información sobre la evaluación de su algoritmo y generar su propio análisis. A continuación se describen los subsistemas de Be4SeD.

5.2.1. Banco de Servicios

Es una colección común de servicios utilizada para evaluar los algoritmos de descubrimiento. Estos servicios son clasificados como *Query* y *Target*. La evaluación se realiza entre un número definido de *servicios Query* y todos los *servicios Target* (1:N). Vale la pena aclarar que los servicios *Query* están incluidos como servicios *Target*. Finalmente, se resalta que no es posible generar evaluaciones entre parejas de servicios *Target* y mucho menos generar evaluaciones de parejas repetidas.

5.2.2. Benchmark de Referencia

Para garantizar que la evaluación de algoritmos de emparejamiento de servicios sea objetiva y confiable, se necesita una referencia común, que proporcione una "verdad absoluta" producto, del criterio de expertos en el tema de Descubrimiento de Servicios. Por esta razón, el *Benchmark de Referencia* constituye un aspecto clave en esta investigación. Éste subsistema, representa los datos de referencia ante los cuales los algoritmos de emparejamiento comparan sus resultados de selección de servicios con el fin de determinar la eficacia de las técnicas de recuperación empleadas. El *Benchmark de Referencia* es creado considerando las siguientes políticas:

5.2.3. Políticas de Creación del Benchmark de Referencia

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

a. Evaluación del emparejamiento por usuario:

$$EMu = \sum_{i=1}^{n} (Wi * Sui) \quad (1)$$

Donde: **Wi** (es el peso asignado por el evaluador según el grado de relevancia del atributo del servicio evaluado), **Sui** (Calificación) y **n** (cantidad de atributos a considerar)

El valor *EMu* es la similitud estimada por un usuario para una pareja de servicios.

La media de similitud para cada uno de los servicios evaluados es la siguiente:

b. Evaluación Total:

$$TE(EM) = \frac{\sum_{u=1}^{n} EMu}{n} \quad (2)$$

Donde **EMu** es la similitud de una comparación y **n** es el número de evaluadores.

5.2.4. Benchmark del Algoritmo

Es la colección de los resultados de recuperación de servicios obtenidos, de la ejecución de los algoritmos de emparejamiento a evaluar. Dichos algoritmos son ejecutados sobre las parejas de *servicios Query* y todos los servicios *Target* (1:N), contenidas en el *Banco de Servicios*. Se debe resaltar que, los resultados de recuperación varían según las técnicas empleadas para determinar la similitud entre servicios.

5.2.5. Análisis estadístico

El desempeño general del sistema se establece utilizando las medidas: *recall* (*r*)*, precisión* (*p*)*, overall* (*o*)*, top-k precisión* (P_k) *y P-presición* (P_p). Para evaluar la calidad del algoritmo de recuperación, se comparan los servicios (*P*) retornados por el *Algoritmo* con los servicios (R) obtenidos en el *Banco de Servicios*. De esta forma se puede determinar un conjunto de verdaderos positivos (*I*), servicios correctamente identificados; igualmente se determina un conjunto de falsos positivos, servicios falsos recuperados (*F = P/I*), y falsos negativos, es decir servicios relevantes no recuperados (*M = R/I*) (Corrales et al., 2008).$Retrel_k$ es el conjunto de servicios relevantes para un *top k* de servicios recuperados, mientras *Rel-p* determina cuantos de los servicios de $Retrel_k$ están en la misma posición del ranking de referencia del *Banco de Servicios* (Zhang et al., 2004). Con base en la cardinalidad de estos conjuntos se tiene:

$$p = \frac{|I|}{|P|} \tag{3}$$

$$r = \frac{|I|}{|R|} \tag{4}$$

$$o = r * \left(2 - \frac{1}{p}\right) \tag{5}$$

$$p_k = \frac{|Retrel_k|}{k} \tag{6}$$

$$p_p = \frac{[Retrel_k]Rel - p}{k} \tag{7}$$

La medida *precisión* estima la fiabilidad de los servicios relevantes recuperados por el algoritmo, en tanto *recall* especifica el porcentaje de los servicios relevantes recuperados. Por su parte, la medida *overall* 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 los servicios empleados en el *Banco de Servicios*. Para estimar la *precisión* y el *recall* de todo el sistema, se emplean los métodos *macro-promedio* y *micro-promedio*(Lewis, 1992), así:

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

$$P = \frac{\sum_{i=1}^{n} p_i}{n} \tag{8}$$

$$R = \frac{\sum_{i=1}^{n} r_i}{n} \tag{9}$$

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.

$$P = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FP_i)} \quad (10)$$

$$R = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} (TP_i + FN_i)} \quad (11)$$

Donde: *TPi*: Son los verdaderos positivos, *FPi*: Son los Falsos positivos, *FNi*: Son los Falsos Negativos.

A partir de los resultados entregados por éste módulo, se realiza un completo análisis estadístico que permite determinar la calidad de un algoritmo de emparejamiento de servicios. Este análisis se fundamenta en la evaluación de las siguientes relaciones: *Precision vs. Recall, Overall, K-Precision vs. K, P-Precision vs. K,* aplicadas en tres escenarios: i) evaluación de las medidas de desempeño comparando servicios de entrada contra los de un mismo dominio contenidos en el repositorio, ii) comparación de los servicios de entrada contra aquellos almacenados que pertenecen a un dominio diferente, y iii) comparación de los servicios de consulta contra todos los servicios contenidos en el repositorio.

5.3. Prototipo de la Plataforma Be4SeD

La Figura 22, expone el diagrama de despliegue de la plataforma Be4SeD. Su implementación fue realizada sobre Glassfish V2.1, con J2EE (versión 1.4), utilizando el *Contenedor de EJB* (*Entrerprise Java Beans, versión 2.1*) para desplegar la lógica de negocio. La interfaz de usuario y los servicios web se encuentran en el *Contenedor Web.* Adicionalmente, se utilizó PostgresSQL (versión 8.3) como motor de base de datos. El *Banco de Servicios,* utiliza el repositorio de procesos de negocio presentado en Vanhatalo et al. (2006).

5.3.1. Banco de Servicios

Está conformado por actividades básicas descritas en una colección de documentos BPEL, encontrados en el repositorio de procesos de negocio presentado en Vanhatalo et al. (2006.), el cual soporta el almacenamiento y consulta de documentos BPEL (y otros documentos XML). Éste provee un API Java para la manipulación de sus archivos como objetos y guarda las descripciones de 53 actividades básicas BPEL, agrupadas en 5 dominios: Vacaciones, Compras, Pagos, Disponibilidad de productos e Información de productos, clasificadas como *Actividades Query* (28) y *Actividades Target* (25). Obteniendo de ellas 1106 parejas, número considerable a evaluar, conformando un gran banco de datos. Finalmente es importante resaltar que a las actividades básicas abstraídas de BPEL, se les adicionó un parámetro de contexto, con el fin de mostrar la capacidad de adaptación del *Banco de Servicios*, permitiendo así la adopción de nuevos atributos y diferentes tipos de representación de servicios. Ésta es una característica muy importante de la plataforma, ya que permite extender su funcionalidad a diferentes dominios de aplicación, como los entornos Web y ambientes de computación ubicua.

5.3.2. Conversor BPEL-Grafos

Transforma las descripciones de comportamiento (BPEL) en su equivalente en grafos, implementando la estrategia presentada por Mendling & Ziemann (2005). 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(Corrales, 2008).

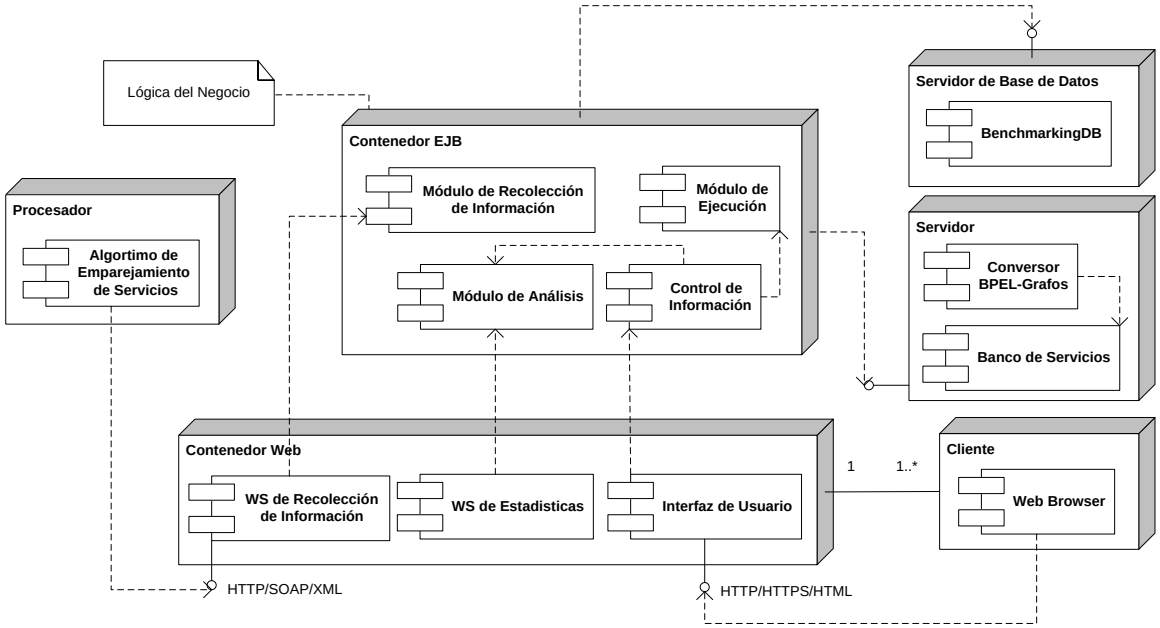


Figura 22. Diagrama de Despliegue de la plataforma Be4SeD

5.3.3. Control de Información

Es el encargado de procesar las peticiones de la interfaz de usuario y encontrar los datos que ésta necesita. Por lo tanto, éste módulo toma información tanto del banco de servicios como de los módulos de análisis y ejecución.

5.3.4. BenchMarkingDB

Almacena las valoraciones realizadas por los expertos en un formato relacional.

5.3.5. Interfaz de Usuario

Facilita la interacción de los expertos con la plataforma. Su lógica de presentación es implementada en el *Módulo de Interfaz de Usuario*. En la Figura 23, se muestra la vista que permite al experto seleccionar el dominio y la actividad a evaluar para posteriormente

construir su *benchmark*. En la Figura 24,se observa la evaluación hecha entre dos nodos *Query* y *Target*. El evaluador realiza la comparación de las actividades asignando una calificación a cada uno de los atributos, según el nivel de similitud. El valor de la calificación a asignar está entre 0 y 5, donde 0 es la mínima y 5 la máxima similitud. Además, el evaluador, al momento de registrarse en Be4SeD, fija un peso de acuerdo a la importancia de cada uno de los atributos. La suma de todos los pesos debe ser igual a 100%.

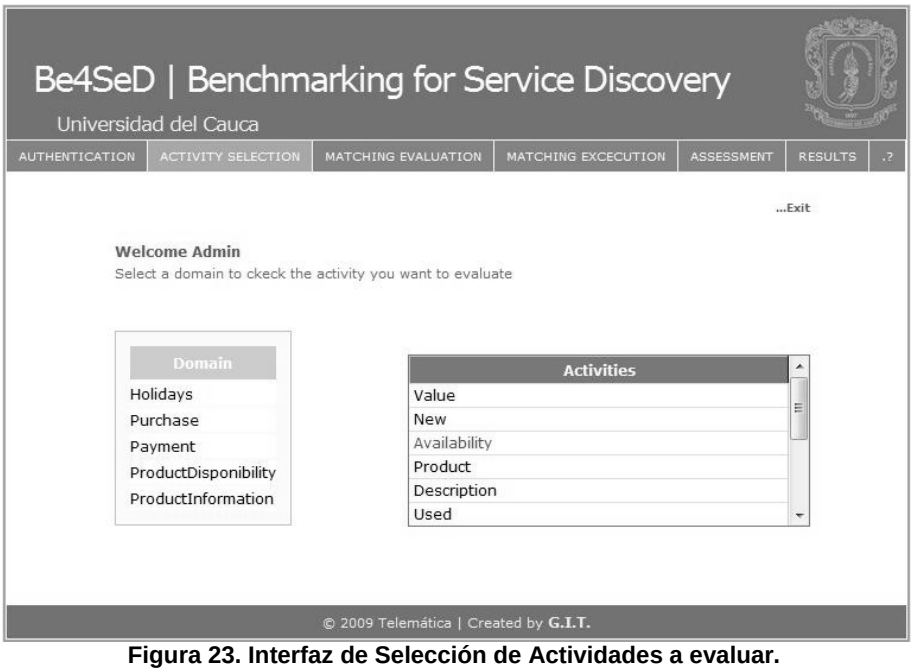


Figura 23. Interfaz de Selección de Actividades a evaluar.

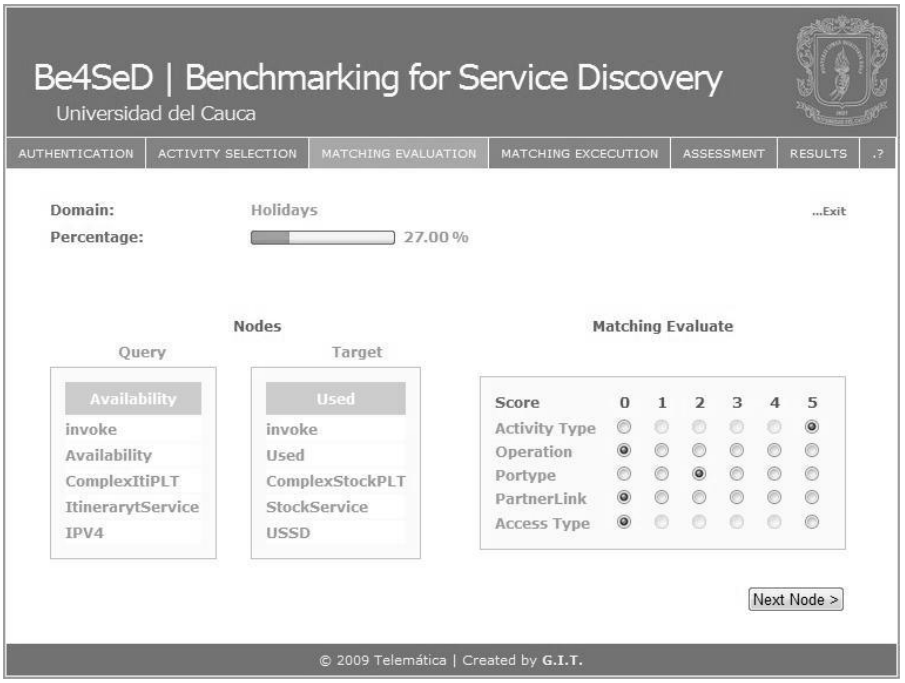


Figura 24.Interfaz de Evaluación.

5.3.6. Módulo de Ejecución

Implementa las políticas para la generación del *Benchmark de Referencia*. El administrador lo ejecuta, una vez los expertos completen la valoración del *Banco de Servicios*. Sin embargo, es importante aclarar que éste módulo, no es el encargado de realizar análisis sobre los datos obtenidos en éste proceso.

5.3.7. Módulo de Recolección de Información

Se encarga de obtener los resultados de algoritmos de emparejamiento de servicios, generando el *Benchmark del Algoritmo*. Para esto, el módulo de recolección de información implementa la lógica que soporta las siguientes operaciones: *Autenticación* - valida el ingreso de datos a la plataforma. Como parámetros de entrada recibe un *login* y un *password*. Retorna una cadena de caracteres (serial) utilizada para ingresar nuevos datos al sistema. *Obtener atributos de las parejas de actividades* - retorna el valor de un parámetro específico de los servicios que conforman la pareja consultada. Como parámetros de entrada recibe el serial, el identificador de la pareja (0-1105), nodo (“a” o “b”) y atributo (0-(n-1)). Para éste caso las actividades cuentan con 5 atributos, n=5. *Activity Type, Operation, Portype, PartnerLink* y *Access Type. Evaluar similitud* - almacena el resultado de la comparación de la pareja evaluada por el algoritmo. Los parámetros de entrada son: serial, identificador de la pareja y la similitud (*Score*). Las anteriores operaciones son expuestas por medio del *WS de Recolección de Información*, facilitando su consumo por parte de las implementaciones de distintos algoritmos.

5.3.8. Módulo de Análisis

Es el encargado de entregar los datos estadísticos generados como resultado de la comparación entre el *Benchmark del Algoritmo* y el *Benchmark de Referencia*. Estos datos, son una adaptación de medidas propias del campo de descubrimiento de servicios, a una metodología de *benchmarking*, demostrando su flexibilidad y por ende la gran utilidad de ésta en diversos entornos. Este componente implementa la lógica de operaciones que retornan los valores de *Precision, Recall, Overall, P-presision y k-precision* utilizando las técnicas de *Macro-promedio* y *Micro-promedio*. También contiene una operación de autenticación similar al expuesto en el *Módulo de Ejecución*. El usuario (Figura 19) puede acceder a esta información consumiendo el *WS de Estadísticas*, que expone las operaciones descritas. Lo anterior, permite adaptar la plataforma Be4SeD a una aplicación externa que utilice estos datos estadísticos para generar un análisis sobre la calidad de su algoritmo. Con la información generada por el *Módulo de Análisis*, la respectiva *Interfaz de Usuario* de Be4SeD permite visualizar una lista ordenada de los resultados arrojados por el algoritmo evaluado (*Benchmarking del Algoritmo*) y la evaluación hecha por los expertos (*Benchmarking de Referencia*), para todas las actividades de consulta contenidas en el *Banco de Servicios*, ver Figura 25. Esta es una característica muy importante de la plataforma Be4SeD. La columna a la izquierda expone el ranking de servicios del *Benchmark de Referencia,* el nodo *Query* es *Payment,* y el *K es* igual a 5. Como se presentó, los *servicios Query* están incluidos como *servicios Target*, es por ello que el primer nodo encontrado en la lista es el mismo nodo *Payment*, el cual obviamente posee la máxima similitud; mostrando que el *Benchmarking de Referencia* es confiable, para determinar la calidad de algoritmos de emparejamiento de servicios.

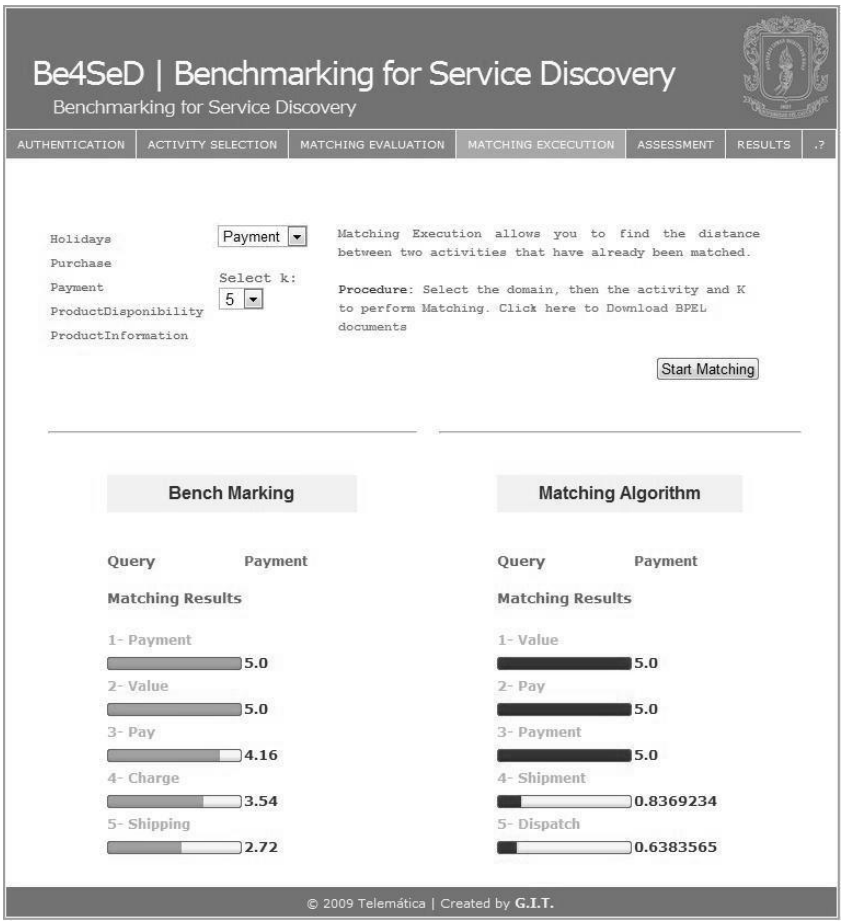


Figura 25. Ranking de Servicios.

5.4. Criterios de Evaluación

Con el fin de asegurar y garantizar el correcto funcionamiento de la plataforma U-*ServiceMatch*, se realizaron pruebas que permiten determinar la calidad de los resultados (eficacia) y el rendimiento la plataforma (eficiencia). Es importante mencionar que el rendimiento del sistema está estrictamente ligado con el hardware del equipo que realiza la función de servidor, ya que éste determina el desempeño del mismo.

Para los módulos y componentes funcionales relevantes dentro de la plataforma U-*ServiceMatch*, se definieron pruebas enfocadas a la evaluación de dos aspectos fundamentales: la calidad de los resultados y el rendimiento de la plataforma. El primero, evalúa la eficacia de los algoritmos, que intervienen en el proceso de recuperación y emparejamiento de la plataforma, y el segundo evalúa los tiempos de respuesta de las solicitudes realizadas a la plataforma.

Para obtener los tiempos de respuesta de los módulos funcionales que conforman la plataforma U-ServiceMatch, se usó la clase nativa de java conocida como *System*.

Finalmente la evaluación y clasificación de los resultados de las pruebas, se realizó a partir de un conjunto de reglas, que determinan los criterios de tiempos de respuesta para aplicaciones que están directamente relacionadas a las características cognitivas de los humanos (Joines, et al., 2002):

- Los usuarios no notan un retardo de menos de 0.1 segundos. Por tanto, un tiempo de respuesta que esté bajo el umbral de 0.1 segundos se clasifica como óptimo.

- Un tiempo de respuesta de menos de 1 segundo no interrumpe el flujo del pensamiento de un usuario, pero deja notar un retardo. Por tanto, un tiempo de respuesta que esté bajo el umbral de 1 segundo se clasifica como bueno.

- Los usuarios aún esperarán la respuesta si está por debajo del umbral de 10 segundos, pero el retardo es notorio. Por tanto, un tiempo de respuesta que esté bajo el umbral de 10 segundos se clasifica como aceptable.

- Después de 10 segundos, los usuarios pierden la concentración y no continúan esperando la respuesta del sistema. Por tanto, un tiempo de respuesta que esté por encima del umbral de 10 segundos se clasifica como deficiente.

5.5. Plan de Pruebas

Las pruebas de calidad y rendimiento, se hacen sobre los módulos que realizan las funcionalidades más importantes y críticas de la plataforma U-ServiceMatch, estas son: Transformación de los procesos BPEL almacenados en el *Repositorio de Servicios*, Recuperación de Actividades Básicas BPEL (*ServiceMatch*) y Recuperación de Actividades Básicas BPEL adaptado al contexto (*SeMatch-Context*).
A continuación en la Tabla 2 se describen las pruebas de eficacia y eficiencia realizadas a las aplicaciones software que conforman la plataforma U-ServiceMatch.

PLAN DE PRUEBAS

Transformación de los procesos BPEL almacenados en el *Repositorio de Servicios*	**Prueba de rendimiento TR1**: Determina el tiempo que tarda el Analizador BPEL-Grafos en transformar los procesos de negocio BPEL, encontrados en el Repositorio de Servicios, en Grafos, y además, obtener, clasificar y organizarlos nodos actividad definidos en él.
Recuperación de Actividades Básicas BPEL (*ServiceMatch*)	**Prueba de calidad de los resultados EC1:** Determina la calidad de los resultados arrojados por la aplicación *ServiceMatch*, y por ende la eficacia de los algoritmos que intervienen en el proceso de emparejamiento. **Prueba de rendimiento ER1**: Determina el tiempo que tarda el módulo de recuperación, en encontrar los servicios más similares para un determinado requerimiento del usuario.
Recuperación de Actividades Básicas BPEL adaptado al Contexto (*SeMatch-Context*)	**Prueba de calidad de los resultados CC1:** Determina la calidad de los resultados arrojados por la aplicación *SeMatch-Context*, y por ende la eficacia de los algoritmos que intervienen en el proceso de emparejamiento, los cuales consideran el contexto, tanto de los requerimientos del servicio, como las restricciones del usuario.

	Prueba de rendimiento CR1: Determina el tiempo que tarda el módulo de recuperación en encontrar los servicios más similares, para una determinada solicitud, considerando el contexto del usuario.
Obtención del Contexto del Dispositivo Móvil	**Prueba de rendimiento OR1:** Determina el tiempo que tarda la plataforma en ejecutar el proceso de obtención de las capacidades de los dispositivos que accede al sistema. Para éste caso, la característica que se obtiene es el tipo de acceso que soporta el cliente móvil.
Recuperación de Servicios A través de Terminales Móviles	**Prueba de rendimiento RR1:** Determina el tiempo que tarda la plataforma, en ejecutar el proceso de recuperación de los servicios pertinentes a una consulta dada a través de diferentes dispositivos móviles.

Tabla 2. Plan de Pruebas

5.6. Especificaciones Técnicas del Servidor Central y de los Dispositivos móviles usados en las pruebas.

Servidor Central

Procesador	RAM	Disco Duro	Sistema Operativo
Genuine Intel(R) T2080 @ 1.73GHZ x2	1 GB	120 GB	Microsoft Windows XP

Tabla 3. Especificaciones técnicas del Servidor Central de U-ServiceMatch

Dispositivos Móviles

Dispositivo	Procesador	RAM	ROM	Tamaño de pantalla	Sistema Operativo
Pocket PC DELL AXIM x51v	Intel PXA270, 520 MHz	64MB	256MB	480 X 640 Pixeles.	Microsoft Windows Mobile 5.0
Nokia N93	Dual ARM 11 332 MHz	64MB	50 MB	128 X 160 Pixeles.	Symbian 9.1
Nokia 6212 NFC	Emulador Series 40 6th Edition SDK				
Nokia 6267	Emulador Series 40 5th Edition SDK				

Tabla 4. Especificaciones técnicas de los dispositivos usados para las pruebas.

5.7. Resultados

A continuación se presentan las gráficas asociadas a los resultados de las pruebas de calidad (eficacia) y rendimiento (eficiencia) definidas en la sección 5.4.1, las tablas donde se consignan los datos de tiempos de respuesta y calidad de los resultados se encuentran en el **ANEXO D.**

- **Transformación de los procesos BPEL almacenados en el Repositorio de Servicios.**

Prueba de rendimiento TR1: en ésta prueba se desea determinar cómo es el rendimiento de la etapa de transformación de las descripciones BPEL publicadas en el repositorio, y por ende, evaluar el mecanismo de consulta implementado sobre el repositorio de procesos. Para esto se cuenta con 119 archivos BPEL publicados en el Repositorio de Servicios (Vanhatalo y otros, 06), para un total de 1008 nodos o actividades básicas.

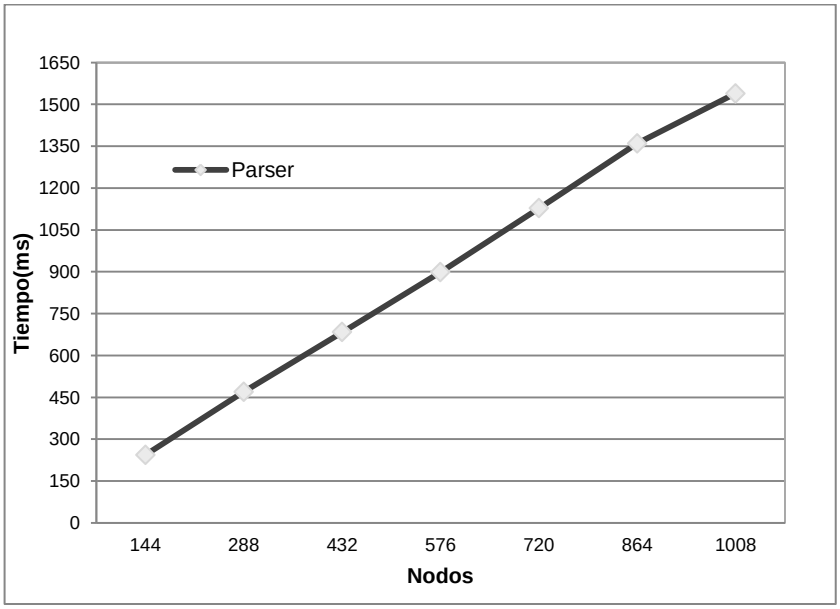


Figura 26. Rendimiento del proceso de Transformación de los documentos BPEL del Repositorio de Servicios en Grafos, además de obtener, clasificar y organizar los nodos por tipo de actividad.

La Figura 26 presenta los tiempos de respuesta de la ejecución del proceso de transformación de BPEL a Grafos. Para lo cual se concluye que tiene un comportamiento lineal y directamente proporcional a la cantidad de nodos contenido en el Repositorio de Servicios. Éste comportamiento es esperado y el tiempo de respuesta es bueno ya que el proceso de transformación de BPEL a Grafos es una tarea muy compleja.

- **Recuperación de actividades básicas BPEL (*ServiceMatch*)**

Prueba de calidad de los resultados EC1: 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 medida de precisión con respecto al número de actividades recuperadas por el algoritmo es presentada en la Figura 27. 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 similitud. De acuerdo a la medida *Total* el mejor desempeño del sistema se obtiene para valores de similitud superiores a 4,41(Figura 30), rango en el cual la precisión oscila entre 0,7 y 0,9. Tomando éste rango como referencia, el k óptimo del sistema se encuentra entre 1 y 5 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.

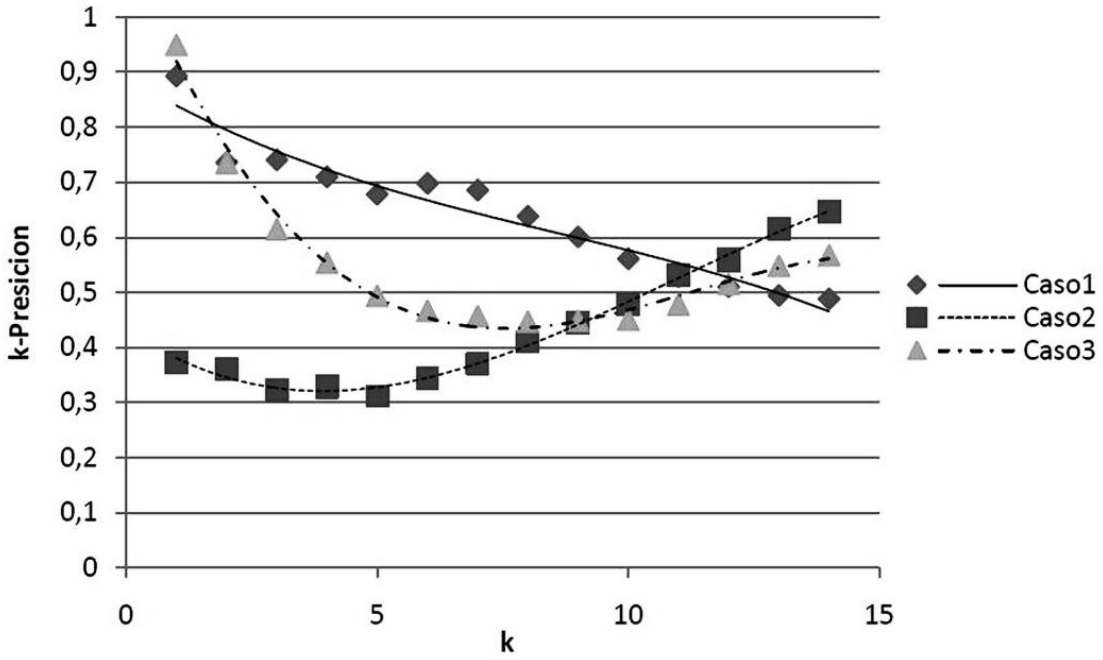


Figura 27. K-Precisión vs. K

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 28, decrecientes a medida que se incrementa el número de actividades k, comportamiento presentado en los tres casos. Las mejores precisiones 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.

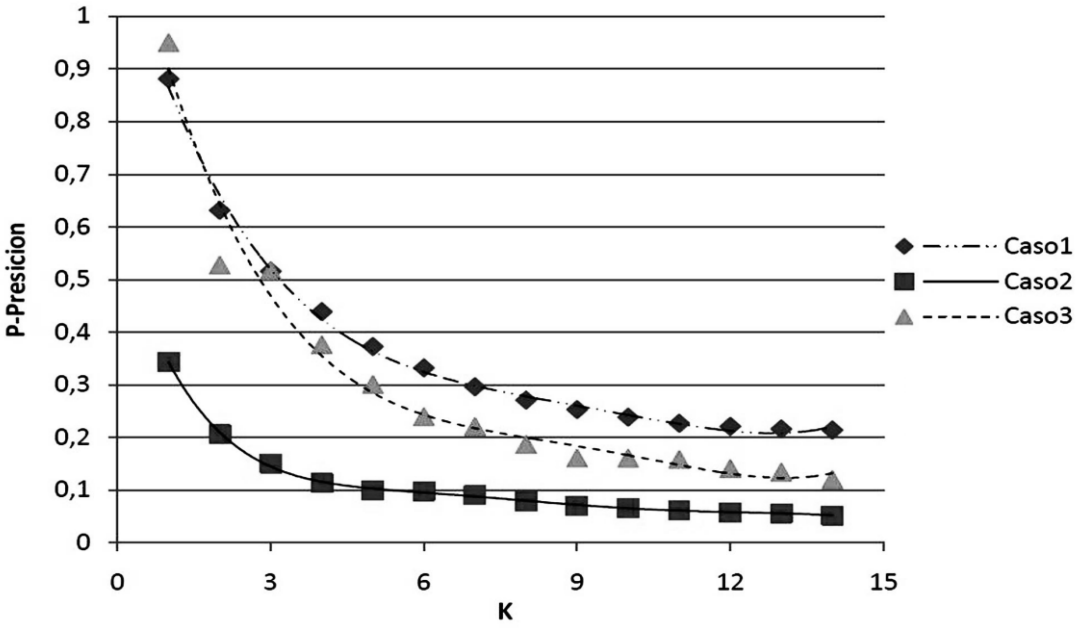


Figura 28. P-Precisión vs. K

En la Figura 29, se puede observar la relación entra las medidas de *precisión* y *recall* para los tres escenarios de prueba. En el caso 1, se tiene una mejor relación entre la medida de *precisión* 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 otros dominios.

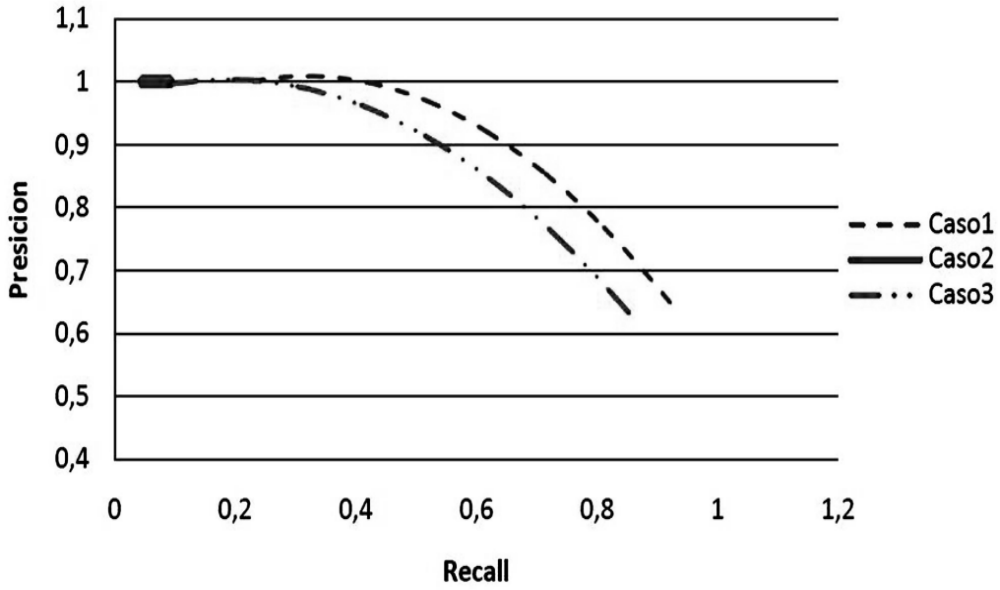


Figura 29. Precisión vs. Recall

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*.

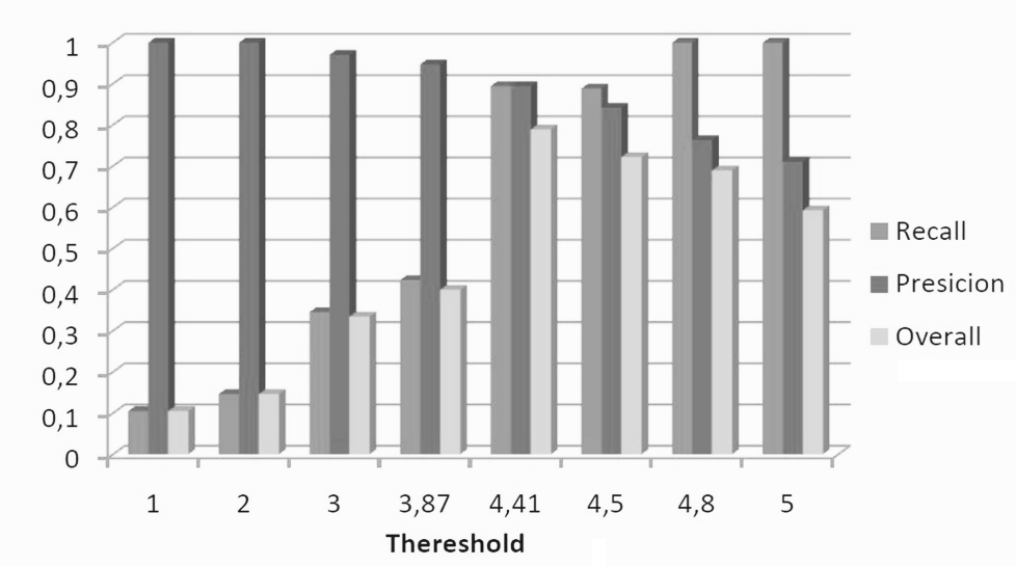


Figura 30. Umbral del Sistema

El desempeño total del sistema se puede apreciar en la Figura 310, donde se identifica el umbral de similitud óptimo para el algoritmo de emparejamiento, equivalente a 4,41, punto en el cual la medida *total* alcanza su tope en un valor cercano al 80%. Igualmente se aprecia que en los valores de similitud 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 similitud, 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 similitud está por encima de 4.

Para las anteriores pruebas se implementó un sistema de recuperación de servicios sobre repositorio de documentos BPEL, el cual almacena 17 archivos BPEL y cuenta con un total de 144 actividades básicas BPEL. Se evaluaron las medidas de *precisión, recall, total, Top-k precisión, P-Precisión* 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*.

Prueba de rendimiento ER1: en esta parte, se presenta los resultados del estudio del tiempo de ejecución del algoritmo de matching de actividades básicas BPEL. Para esta medida se cuenta con 17 archivos BPEL publicados en el *Repositorio de Servicios* (Vanhatalo y otros, 06), para un total de 144 nodos o actividades básicas. Además, se tomó 5 documentos *Query*, conformados por actividades básicas, denominadas *nodos Query* que serán comparados con los *nodos Target*, encontrados en el *Repositorio de Servicios* (ver Tabla 4).

BPEL Query Files	Actividades Básicas
Holidays	5
Payment	4
ProductDisponibility	7
ProductInformation	7
Purchase	7
TOTAL	30

Tabla 5. Archivos Query BPEL

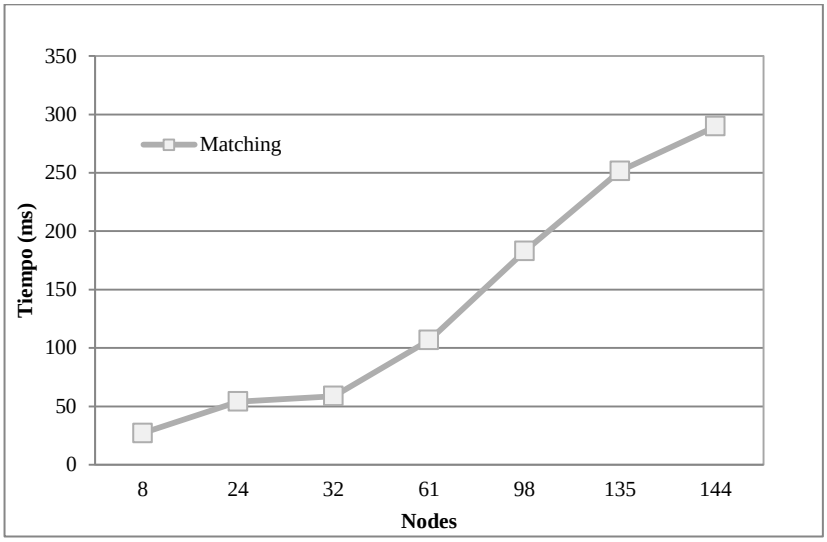


Figura 31. Rendimiento del proceso de Matching de las actividades básicas BPEL

De la Figura 31, se puede concluir que el rendimiento del sistema es proporcional al número de documentos BPEL publicados en el repositorio de servicios. Específicamente, el tiempo de respuesta de nuestro algoritmo de emparejamiento, varía con relación a los servicios publicados en el repositorio. El *Transformador BPEL – Grafos* presenta éste mismo comportamiento, a medida que los nodos de entrada del sistema se van incrementando, el tiempo también lo hace. Éste comportamiento es esperado, ya que el proceso de transformación de BPEL a grafos no es tan simple.

Luego se linealizó los resultados de rendimiento del proceso de Matching, generalizando su comportamiento en función del número de nodos consultados, y de esta manera se evaluó su rendimiento, de acuerdo con lo expuesto en (Joines, et al., 2002). Así, se determinó que *SeMatch-Context* tiene un comportamiento:

- **Óptimo** (tiempo de respuesta menor o igual a 0,1s.): cuando el número de nodos comparados es menor o igual a 51,2.
- **Bueno** (tiempo de respuesta entre 0,1s. y 1s.): cuando el número de nodos comparados es mayor que 51,2 nodos y menor que 527,9.
- **Aceptable** (tiempo de respuesta entre 1s. y 10s.): cuando el número de nodos comparados es mayor que 527,9 nodos y menor que 5294,8.

- **Deficiente** (tiempo de respuesta mayor a 10s.): cuando el número de nodos comparados es superior a 5294,8.

❖ **Recuperación de actividades básicas BPEL adaptado al contexto (*SeMatch-Context*)**

Prueba de calidad de los resultados CC1: como se puede observar los resultados de ésta prueba no difieren sustancialmente de la información obtenida en la Prueba EC1. Manteniendo los siguientes datos: el mejor desempeño del sistema aún se obtiene para valores de similitud superiores a 4,41(Figura 30 y Figura 32d), rango en el cual la precisión oscila entre 0,7 y 0,9. Tomando éste rango como referencia el k óptimo del sistema se encuentra entre 1 y 5 actividades recuperadas. 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 (Figura 28 y Figura 32a). Existe una diferencia en el comportamiento del caso 1 de la Figura 32b con respecto a la Figura 27. En la Figura 27, éste caso tiene un comportamiento ideal, dado a la naturaleza del caso 1(los servicios corresponden al mismo dominio). Sin embargo, en el caso 1 de la Figura 32b,el comportamiento es similar al caso 3, dado que la adición de otro parámetro en la búsqueda bajo el mismo dominio de consulta disminuye la probabilidad de encontrar las actividades más similares a una actividad requerida, debido al ruido que agregan las actividades con diferentes tipos de acceso.

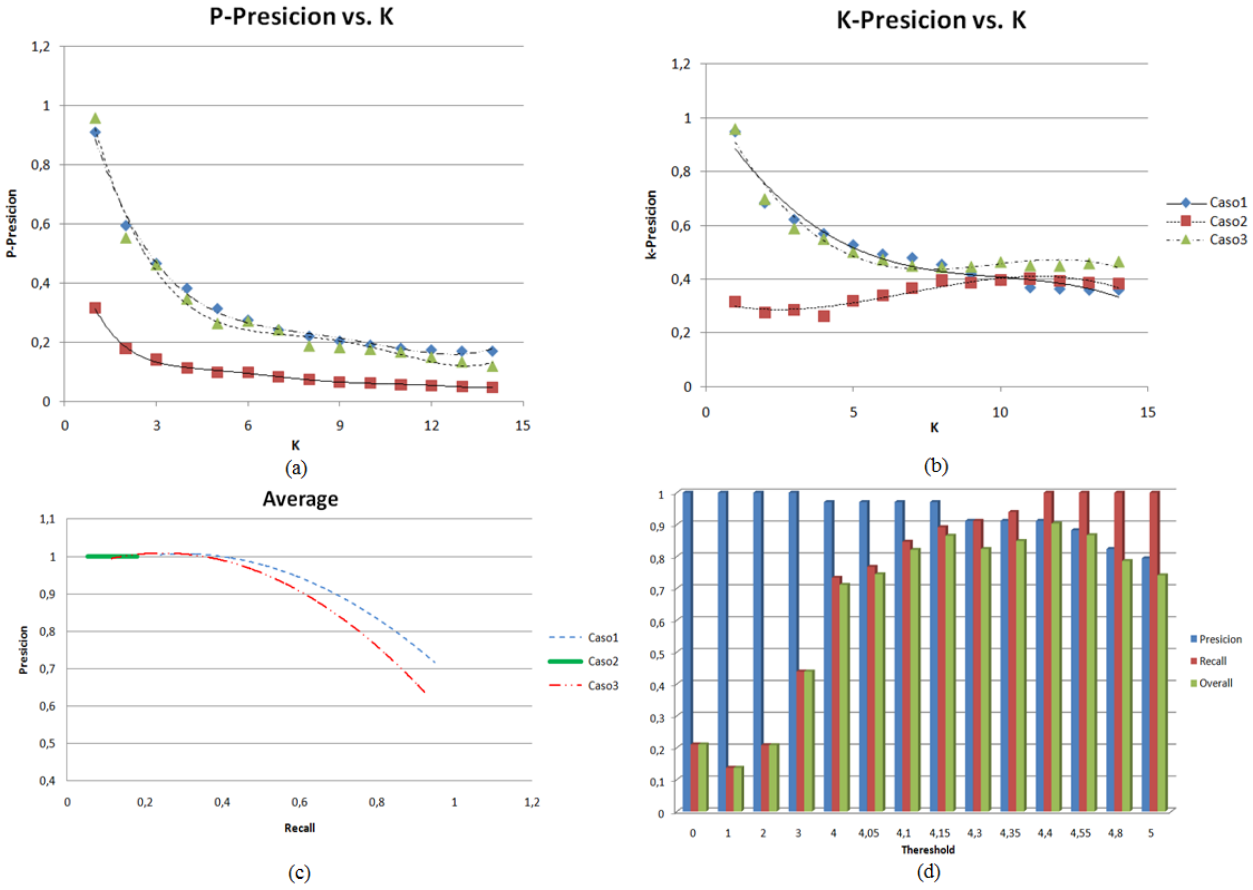


Figura 32. . Medidas de Desempeño para la aplicación SeMatch-Context

Prueba de rendimiento CR1: como se puede observar en la Figura 33 el tiempo de respuesta para el proceso de matching a nivel atómico considerando el contexto posee el mismo comportamiento y tiempo de ejecución es similar a la prueba de rendimiento ER1, siendo estos dos tiempos, proporcionales al número de nodos consultados.

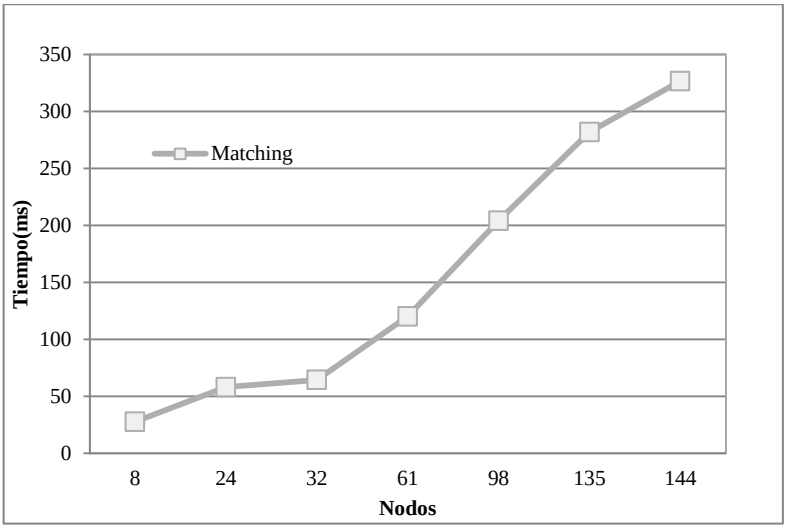


Figura 33. Rendimiento del proceso de Matching de las actividades básicas BPEL considerando el contexto.

Pero existe un incremento en el tiempo de respuesta de *SeMatch-Context*, éste incremento posee una parte constante, correspondiente al tiempo que toma la identificación del contexto del dispositivo móvil, y otra variable, proporcional al número de actividades sometidas al proceso de descarte por contexto.

Luego se linealizó los resultados de rendimiento de *SeMatch-Context*, generalizando su comportamiento en función del número de nodos consultados, y de esta manera se evaluó su rendimiento de acuerdo con lo expuesto en (Joines, et al., 2002). Así, se determinó que *SeMatch-Context* tiene un comportamiento:

- **Óptimo** (tiempo de respuesta menor o igual a 0,1s.): cuando el número de nodos comparados es menor o igual a 46,1.
- **Bueno** (tiempo de respuesta entre 0,1s. y 1s.): cuando el número de nodos comparados es mayor que 46,1 nodos y menor que 466,4.
- **Aceptable** (tiempo de respuesta entre 1s. y 10s.): cuando el número de nodos comparados es mayor que 466,4 nodos y menor que 4669,8.
- **Deficiente** (tiempo de respuesta mayor a 10s.): cuando el número de nodos comparados es superior a 4669,8.

❖ **Obtención del Contexto del Dispositivo Móvil**

Prueba de rendimiento OR1: en la Figura 34, se expone los resultados del rendimiento de la plataforma en el proceso de obtención de las capacidades de dispositivos, para nuestro caso: *Tipo de Acceso* (creación del contexto de entrega). El acceso a la plataforma se realiza desde diferentes terminales móviles a través de una red Wi-Fi y diferentes emuladores.

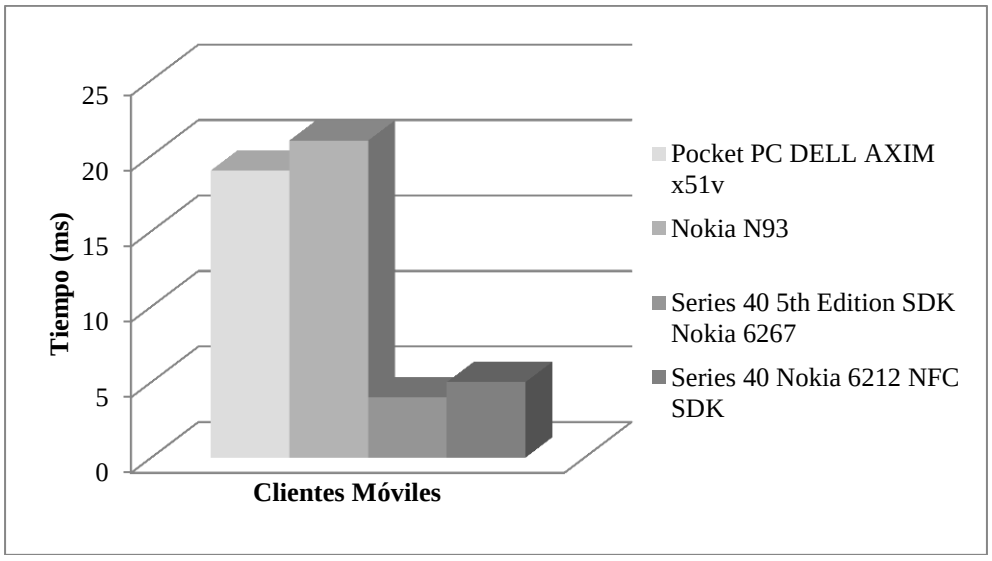


Figura 34. Gráfica de rendimiento en el proceso de creación del contexto de entrega.

En los cuatro dispositivos utilizados, dos de ellos emuladores (Series 40 5th Edition SDK Nokia 6267 y Series 40 Nokia 6212 NFC SDK), presentan un tiempo de respuesta bueno en el proceso de obtención del contexto de entrega. Obviamente el rendimiento para los emuladores es notablemente inferior, dado que, tanto la plataforma como el cliente residen en el mismo servidor.

❖ Recuperación de servicios a través de terminales móviles

Prueba de rendimiento RR1:

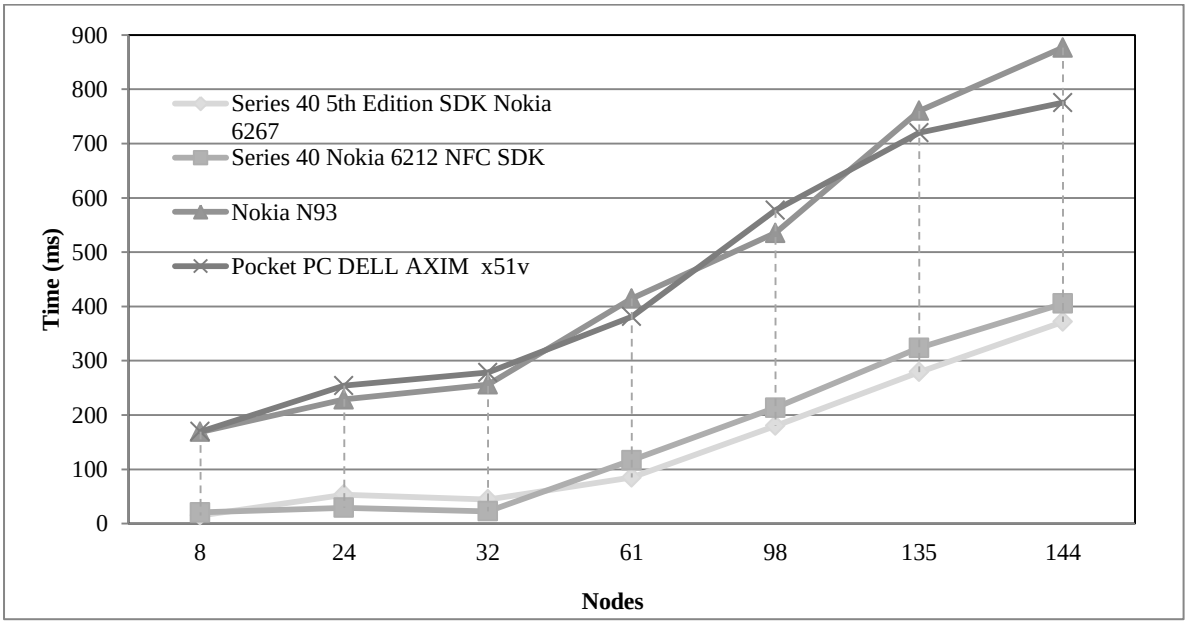


Figura 35. Gráfica de rendimiento en el proceso de recuperación de servicios sobre diferentes terminales móviles para la aplicación SeMatch-Context.

La Figura 35, presenta los tiempos de ejecución de la aplicación *SeMatch-Context* utilizada por diversos clientes móviles. Para esta medida se contó con 17 archivos BPEL publicados en el *Repositorio de Servicios* (Vanhatalo y otros, 06), para un total de 144 nodos o actividades básicas. Además, se tomó 5 documentos *Query* (ver Tabla 4), conformados por actividades básicas, *nodos Query,* que fueron comparados con los 144 *nodosTarget*. Finalmente, los resultados obtenidos para los 5 archivos de consulta BPEL fueron promediados, y así se obtuvo la media del sistema para diferentes servicios de consulta. De estos resultados se concluye que el tiempo de respuesta de *SeMatch-Context*, tiene el mismo comportamiento tanto en dispositivos reales, como en emuladores, siendo proporcional al número de *nodos Target*. Sin embargo se observa una diferencia de tiempo, aparentemente constante, entre las dos respuestas. La razón de ésta diferencia radica en que el emulador se encuentra en el mismo equipo donde se despliega la plataforma, lo cual elimina la latencia provocada por la red de acceso. En cambio, la respuesta en un dispositivo real depende tanto de la latencia de la red, las capacidades de procesamiento (hardware) y la rapidez de las interfaces utilizadas para su acceso.

Luego se linealizó los resultados de rendimiento de *SeMatch-Context* en dispositivos reales, generalizando el comportamiento promedio de los dos dispositivos reales utilizados para esta experimentación, en función del número de nodos consultados. De esta manera se evaluó su rendimiento de acuerdo a lo expuesto en (Joines, et al., 2002). Así, se determinó que *SeMatch*-Context, accedido desde dispositivos reales, tiene un comportamiento:

- **Óptimo** (tiempo de respuesta menor o igual a 0,1s.): cuando el número de nodos comparados es menor o igual a -2,7, éste valor nos indica que el sistema no puede dar éste tipo de respuestas.
- **Bueno** (tiempo de respuesta entre 0,1s. y 1s.): cuando el número de nodos comparados es menor que 374,3.
- **Aceptable** (tiempo de respuesta entre 1s. y 10s.): cuando el número de nodos comparados es mayor que 374,3 nodos y menor que 4145,8.
- **Deficiente** (tiempo de respuesta mayor a 10s.): cuando el número de nodos comparados es superior a 4145,8.

5.8. Resumen

En éste capítulo se describió detalladamente Be4SeD, herramienta basada en la metodología de *Benchmarking*, utilizada para evaluar los algoritmos de emparejamiento de la plataforma U-ServiceMatch. Seguido, se expuso las diferentes pruebas realizadas a la plataforma. Éstas permitieron determinarla calidad (eficacia) y el rendimiento (eficiencia) de los subsistemas que componen la plataforma (*ServiceMatch* y *SeMatch-Context*).

En la evaluación de la recuperación de actividades básicas BPEL adaptadas o no al contexto (*ServiceMatch* y *SeMatch-Context*), Be4SeD definió tres escenarios, obteniendo un excelente comportamiento para los tres casos. Concluyendo que si la plataforma no encuentra un servicio que satisfaga exactamente los requerimientos del usuario, los más similares son recuperados y propuestos, prueba EC1.

Para el proceso de transformación de los archivos BPEL a grafos, se obtuvo un comportamiento lineal, proporcional al número de servicios almacenados en el repositorio. El estudio realizado al rendimiento de los subsistemas *ServiceMatch* y *SeMatch-Context*,

muestra un comportamiento similar en el tiempo de ejecución de las dos aplicaciones, proporcional al número de actividades consultadas. Pero existe un incremento en el tiempo de respuesta de *SeMatch-Context*, éste incremento posee una parte constante, correspondiente al tiempo que toma la identificación del contexto del dispositivo móvil, y otra variable, proporcional al número de actividades sometidas al proceso de descarte por contexto. También se determinó que su comportamiento es óptimo cuando la cantidad de nodos consultados es menor a 46.1, bueno, 466.4, aceptable, 4669.8 y deficiente cuando el número de nodos supera éste umbral.

Por último se observó que el tiempo de respuesta de *SeMatch-Context*, tiene el mismo comportamiento tanto en dispositivos reales, como en emuladores, siendo proporcional al número de *nodos Target*. Sin embargo existe una diferencia de tiempo, aparentemente constante, entre las dos respuestas. La razón de ésta diferencia radica en que el emulador se encuentra en el mismo equipo donde se despliega la plataforma, lo cual elimina la latencia provocada por la red de acceso. En cambio, la respuesta en un dispositivo real depende tanto de la latencia de la red, las capacidades de procesamiento (hardware) y la rapidez de las interfaces utilizadas para su acceso. También se determinó que el comportamiento de *SeMatch-Context* en dispositivos reales, es bueno cuando la cantidad de nodos consultados es menor a 374.3, aceptable 4145.8 y deficiente cuando el número de nodos supera éste umbral.

Como conclusión de éste capítulo se presenta la tabla 5, con las formulas resultado de la linealización de las pruebas de rendimiento ER1, CR1 y RR1.

Prueba	Rendimiento Linealizado
ER1: Recuperación de Actividades Básicas BPEL (*ServiceMatch*)	t = (**1,888***r + **3,2369**)*q
CR1: Recuperación de Actividades Básicas BPEL adaptado al Contexto (*SeMatch-Context*)	t = (**2,1411***r + **1,2846**)*q
RR1: Recuperación de Servicios A través de Terminales Móviles	t = (**2,3799***r + **109,16**)*q

Tabla 6. Formulas linealizadas de rendimiento

En la tabla 6, se observa que los módulos muestran un comportamiento lineal, especialmente *SeMatch-Context*; donde r es el número de nodos comparados con la *Query*, y q es el número de consultas. Éste resultado es de gran importancia ya que a partir de él se puede proyectar el rendimiento de un sistema de composición basado en la utilización de la plataforma propuesta.

Capítulo VI

CONCLUSIONES Y TRABAJO FUTURO

La computación Ubicua comienza a ser una realidad gracias fundamentalmente a los avances realizados en el campo de la microelectrónica, que permiten dotar de capacidades de cómputo y comunicación a cada vez más elementos del mundo físico. El reto ahora es aprovechar estas capacidades, generando dinámicamente aplicaciones que permitan integrar los servicios que son expuestos en su entorno.

En esta tesis se ha abordado ese problema, contribuyendo a la definición de parámetros para la adaptación del descubrimiento de servicios, en ambientes de computación ubicua. Inicialmente la investigación se centró en la aplicación del descubrimiento como una etapa previa a la composición de servicios en entornos ubicuos. Su estudio nos llevó a analizar el problema del emparejamiento de actividades básicas y la personalización, para adaptar el descubrimiento a las restricciones que imponen estos entornos. A partir de éste análisis, surgió la propuesta de la plataforma U-ServiceMatch, aplicación que brinda una solución al descubrimiento de servicios en Ambientes Ubicuos.

Éste capítulo recopila las principales contribuciones del presente proyecto de grado, sección 6.1. Además expone las conclusiones extraídas de la realización del mismo, sección 6.2. La sección 6.3 finaliza proponiendo los trabajos futuros.

6.1. Contribuciones

Entre las principales contribuciones de éste proyecto de grado se destacan las siguientes:

- U-ServiceMatch, plataforma producto del análisis de las necesidades que implica la adaptación del descubrimiento de servicios a entornos ubicuos. Ésta plataforma considera características de personalización, tales como Contexto y Perfil de usuario, brindando una solución al descubrimiento de servicios en estos ambientes.

- La definición de técnicas y algoritmos adaptados a las características y requisitos propios de los ambientes de computación ubicua. En éste apartado se resalta el estudio realizado sobre el emparejamiento de actividades básicas y la personalización de los servicios, teniendo en cuenta tanto el perfil de usuario, como el contexto.

- La definición y utilización de repositorios que permiten contextualizar de mejor manera los servicios a descubrir. Para éste fin, se tuvo en cuenta dos repositorios adicionales al de servicios, uno de ellos proporciona datos útiles para el descubrimiento, tomando la información del usuario y el otro identificando las capacidades del dispositivo que solicita el descubrimiento.

- El desarrollo de Be4SeD, plataforma dedicada a la evaluación de técnicas de emparejamiento de servicios. Be4SeD implementa una metodología de *Benchmaking,* en la cual se genera manualmente, un *Benchmark de Referencia* a partir del criterio de varios expertos en emparejamiento de servicios. Esta plataforma entrega un análisis

estadístico basado en medidas como *Recall, precision, Overall, k-Precision y P-Precision.*

- Como contribución final se destaca el soporte que la plataforma U-ServicMatch, brinda al componente de descubrimiento de servicios en la tesis de maestría: **Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua,** desarrollada al interior de la Universidad del Cauca. Así, se da continuación a la investigación y desarrollo de servicios ubicuos, en la comunidad académica de la Universidad, con el objetivo de despertar el interés por estas tecnologías en la comunidad investigativa del país, fortaleciendo el área de Servicios Avanzados de Telecomunicaciones del GIT y del grupo de interés en desarrollo de aplicaciones móviles e inalámbricas W@PCOLOMBIA, a través de la incursión en el desarrollo de sistemas ubicuos, amplio tema de investigación en la actualidad.

6.2. Conclusiones

- Dentro del presente proyecto de grado, se tuvo en cuenta las necesidades que presenta el descubrimiento de servicios en un entorno ubicuo y las falencias existentes en las técnicas propuestas para éste dominio. Encontrando que la principal carencia era la ausencia de una solución que aborde la personalización en el proceso de descubrimiento de servicios, considerando no solo las capacidades del dispositivo, sino también las características y preferencias de usuario.

  Al concluir el trabajo desarrollado en el presente proyecto de grado, se entregó una plataforma de descubrimiento de servicios en ambientes ubicuos. Esta plataforma, a partir de la descripción del perfil de usuario, realiza un proceso de sugerencia de servicios, relacionando los servicios consultados y consumidos por él y/o personas con preferencias similares. También esta plataforma identifica el contexto del usuario, exigiendo el uso de información implícita que relacione tanto las limitaciones/restricciones del solicitante, como los requerimientos del proveedor de servicios, lo cual afecta la calidad y relevancia de los resultados retornados por los mecanismos de recuperación de servicios. Teniendo en cuenta lo anterior, se concluye que el trabajo realizado es un avance en la problemática del descubrimiento de servicios, asociado a la calidad y relevancia de los servicios retornados en ambientes ubicuos.

- Una etapa fundamental en éste trabajo es el emparejamiento de servicios, ya que un óptimo emparejamiento permite recuperar los servicios más adecuados, que se ajusten a las consultas hechas por el usuario. Con relación a esta temática, en el estado del arte se encontró las siguientes falencias asociadas al emparejamiento de servicios en entornos de computación ubicua:

  - Algunas propuestas se basan en comparaciones realizadas por razonadores semánticos, lo cual genera un incremento en el procesamiento, comprometiendo el tiempo de respuesta de un sistema de descubrimiento de servicios.
  - Otras soluciones, realizan un emparejamiento a nivel del comportamiento de procesos de negocio. éste tipo de comparación es muy compleja, y al igual que el uso de razonadores semánticos, incrementa la cantidad de procesamiento requerida para una consulta, aumentando así el tiempo entre solicitud y respuesta.

Teniendo en cuenta las anteriores consideraciones se concluye que en entornos de computación ubicua no es recomendable: realizar un emparejamiento a nivel del comportamiento de procesos de negocio, dada su complejidad computacional, y tampoco el uso de ontologías en la fase de emparejamiento, ya que el proceso de razonamiento incrementa los tiempos de respuesta, factor crítico en estos entornos. Así, el presente trabajo de grado propone un emparejamiento atómico a nivel de actividades básicas BPEL, utilizando una comparación lingüística, basada en diccionarios, para calcular la distancia semántica entre ellas.

- Diversas investigaciones sostienen que utilizar modelos formales para la representación de procesos, es de gran ayuda ya que ofrecen características, como integridad, compatibilidad y paralelismo, importantes en los procesos de composición que soporta el descubrimiento de servicios. Así, del estudio realizado en la sección 3.1.1, se concluyó que el modelo de representación formal de grafos ofrece una simple y madura notación para describir servicios, y permite trasladar el problema de emparejamiento de servicios a un problema matemático de emparejamiento de grafos.

- A partir del estudio de diversos trabajos relacionados con la personalización, se concluyó que la mejor manera de soportarla es a través de la definición de tres repositorios de información: i) **Repositorio de servicios**: el cual soporta el almacenamiento y consulta de documentos BPEL (y otros documentos XML). ii) **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) y WURFL (Wireless Universal Resource) (Passani, 2007). ii) **Repositorio de usuarios**: almacena detalles relacionados con los usuarios, estos incluyen preferencias explicitas provistas por los usuarios o datos implícitos capturados dinámicamente, tales como historial de servicios consumidos o patrones de uso.

  Con el uso de estos repositorios la plataforma desarrollada despliega un descubrimiento personalizado, y soporta dos fases dentro de éste proceso: sugerencia de servicios por perfil de usuario, para reducir el dominio de búsqueda de servicios, y descarte por requerimientos de contexto de servicio, para refinar la búsqueda retornando servicios que puedan ser consumidos por el solicitante. Así, se concluye que la personalización del proceso de descubrimiento de servicios, aumenta el grado de calidad de la búsqueda (proceso de descarte por requerimientos de contexto) y puede mejorar el rendimiento del proceso de descubrimiento de servicios (sugerencia de servicios por perfil de usuario).

- Del estudio de rendimiento realizado a los módulos *ServiceMatch* y *SeMatch-Context* se concluyó que: éstos dos subsistemas, muestra un comportamiento similar en el tiempo de ejecución de las dos aplicaciones, proporcional al número de actividades consultadas. Pero, existe un incremento en el tiempo de respuesta de *SeMatch-Context*, éste incremento posee una parte constante, correspondiente al tiempo que toma la identificación del contexto del dispositivo móvil, y otra variable, proporcional al número de actividades sometidas al proceso de descarte por contexto. También se determinó que su comportamiento es óptimo, cuando la cantidad de nodos consultados es menor a 45, bueno, 453, aceptable, 4533 y deficiente cuando el número de nodos supera éste umbral. Por último se observó que el tiempo de respuesta de *SeMatch-Context*, tiene el mismo comportamiento tanto en dispositivos reales, como en emuladores, siendo

proporcional al número de nodos Target. Sin embargo, existe una diferencia de tiempo aparentemente constante entre las dos respuestas. La razón de ésta diferencia, radica en que el emulador se encuentra en el mismo equipo donde se despliega la plataforma, lo cual elimina la latencia provocada por la red de acceso. En cambio, la respuesta en un dispositivo real depende tanto de la latencia de la red, las capacidades de procesamiento (hardware) y la rapidez de las interfaces utilizadas para su acceso. También se determinó que el comportamiento de *SeMatch-Context* en dispositivos reales, es óptimo cuando la cantidad de nodos consultados es menor a 36, bueno, 358, aceptable 3580 y deficiente cuando el número de nodos supera éste umbral.

- Los resultados entregados por Be4SeD permiten concluir, a través de medidas como precision, *Recall*, *Overall*, *k-precision* y *P-Precision*, que el algoritmo de emparejamiento de servicios utilizado, asegura un cálculo de similitud de gran calidad, apto para ser aplicado en la búsqueda de servicios dentro del proceso de descubrimiento en un entorno ubicuo.

- De la plataforma Be4SeD, herramienta utilizada para la evaluación del algoritmo de emparejamiento, se puede concluir que la filosofía de Benchmarking es muy útil al momento de generar herramientas que permitan evaluar el desempeño de algoritmos de emparejamiento de servicios. La aplicación de esta metodología a técnicas de recuperación llevó a adaptar nuevas medidas, propias de éste campo de estudio, demostrando su flexibilidad y por ente la gran utilidad de esta metodología en diversos entornos. Además, Be4SeD presenta un benchmarking público con fines académicos, orientado a fortalecer el campo del descubrimiento de servicios, brindando la posibilidad que nuevas investigaciones puedan enriquecer éste proceso, logrando la construcción de una guía que permita avanzar de manera más efectiva en pro de perfeccionar los algoritmos de descubrimiento de servicios.

6.3. Trabajos futuros

Ésta tesis ha aportado soluciones al problema del descubrimiento de servicios, adaptando sus algoritmos a las nuevas necesidades y restricciones que imponen los entornos ubicuos. Así, con relación al campo de estudio de éste proyecto de grado se propone los siguientes trabajos futuros:

Estudio y definición de nuevas características que extiendan la descripción del usuario en un entorno ubicuo

El avance de diversos proyectos relacionados con la personalización de sistemas de información (Guerrero, et al., 2010), (Abbar, et al., 2008), permite explorar nuevas dimensiones en la descripción de las características de usuario, aplicadas a la búsqueda de servicios. La adición de estas características puede brindar recursos para soportar nuevas funcionalidades en sistemas de descubrimiento de servicios. Así, se propone continuar éste estudio para encontrar nuevas técnicas que perfeccionen el descubrimiento de servicios en ambientes ubicuos.

Estudio y definición de un sistema de organización de registro de servicios

En algunos trabajos se hace referencia a la organización de registros, sugiriendo la aplicación de políticas de indexación, para reducir el dominio de búsqueda en repositorios de

servicios de considerable tamaño, disminuyendo de éste modo el tiempo entre solicitud y respuesta. Así, se propone como trabajo futuro un estudio sobre sistemas de indexación aplicados a repositorios de servicios en ambientes de computación ubicua.

Estudio y definición de técnicas de emparejamiento basadas en semántica aplicadas a entornos ubicuos.

Actualmente, el mundo académico presenta procesos basados en inteligencia artificial y comparaciones semánticas, que brindan expectativas prometedoras en el campo de la recuperación de información. Su aplicación a entornos de computación ubicua se ha limitado debido a la cantidad de procesamiento que requieren. Pero con los crecientes avances tecnológicos, es posible pensar en una futura aplicación de éste concepto en ambientes de computación ubicua. Es por eso que en éste apartado se propone un estudio de técnicas de emparejamiento basadas en semántica aplicadas a entornos ubicuos.

Evolución de Be4SeD

Como trabajo futuro se propone llevar esta herramienta a un proceso más complejo de benchmarking. Capaz de monitorear la evolución de varias versiones de algoritmos de emparejamiento de servicios, mostrando su utilidad en los procesos de evaluación y mejora de sus enfoques.

Experimentación de U-ServiceMatch en un ambiente real

Por último se propone realizar la experimentación de la plataforma en un ambiente real, para medir el grado de satisfacción de los usuarios, y especialmente determinar su desempeño dentro del proceso de composición de servicios en ambientes de computación ubicua.

Referencias

Abbar S. [et al.] A personalized access model: concepts and services for content delivery platforms [Conferencia] // Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services. - Linz, Austria : ACM New York, NY, USA, 2008. - págs. 41-47.

Almenárez Florina Arquitectura de Seguridad para Entornos de Computación Ubicua Abiertos y Dinámicos [Book]. - Escuela Politécnica Superior : Tesis Doctoral. Departamento de Ingeniería Telemática, 2005.

Álvarez Pedro y Espinoza Mauricio. El ciclo de vida de un servicio Web compuesto: virtudes y carencias de las soluciones actuales. [Informe]. - Zaragoza, España : Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza, 2005.

Andrews Tony [et al.] Business Process Execution Language Version 1.1. the BPEL4WS Specification [En línea]. - 05 de 05 de 2003. - 13 de 05 de 2010. - http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf.

Angell R. C., Freund G. y Willett P. Automatic spelling correction using a trigram similarity measure [Publicación periódica] // Information Processing and management.. - 1983. - Vol. 19. - págs. 255-261..

Arabshian [et al.] Distributed context-aware agent architecture for global service discovery [Conferencia] // 2nd International Workshop on the use of Semantic Web Technologies for Ubiquitous and Mobile Applications (SWUMA'06). - Trentino, Italy : [s.n.], 2006.

Arabshian [et al.] Gloserv: Global Service Discovery Architecture [Conferencia] // Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.. - 2004. - págs. 319 - 325 .

Asfari O. [et al.] Personalized Access to Information by Query Reformulation Based on the State of the Current Task and User Profile [Conferencia] // SEMAPRO. - 2009.

Bandara Ayomi [et al.] A Semantic Approach for Service Matching in Pervasive Environments [Sección del libro]. - [s.l.] : Universidad de Southampton, 2007.

Barkhuus L Ubiquitous Computing: Transparency in Context-Aware Mobile Computing [Book]. - Gothenborg : Doctoral Consortium at UbiComp 2002, 2002.

Barreiro D. [et al.] Personalized Reliable Web service Compositions [Conferencia] // CEUR-WS. - 2008.

Ben Mokhtar S. Semantic Middleware for Service-Oriented Pervasive Computing [Book]. - Paris, Francia : tesis presentada a la Universidad DEVANT L'UNIVERSIT´ E DE PARIS 6 para optar al grado de DOCTEUR DE L'UNIVERSIT´ E DE PARIS 6, 2007.

Ben Mokhtar S., Georgantas N. y Issarny V. Cocoa: Conversation-based service composition for pervasive computing environments [Conferencia] // Proceedings of ICPS. - Lyon, France : [s.n.], 2006. - págs. 29-38.

Ben Mokhtar Sonia, Georgantas Nikolaos y Issarny Valérie Ad hoc Composition of User Tasks in Pervasive Computing Environments [Conferencia] // Proceedings of 4th Workshop on Software Composition (ETAPS'05). - Edinburgh. Scotland : [s.n.], 2005.

Bottaro Andre, Gerodolle Anne y Lalanda Philippe Pervasive Service Composition in the Home Network. Advanced Information Networking and Applications [Sección del libro]. - 2007.

Branca A., Stella E. y Distante A. Qualitative scene interpretation using planar surfaces. [Publicación periódica] // Autonomous Robots. - 8(2):129-139 : [s.n.], 2000.

Bunke H. Graph matching: Theoretical foundations, algorithms, and applications [Book Section]. - [s.l.] : pages 82–88, 2000.

Christmas W.J., Kittler J. y Petrou M. Structural matching in computer vision using probabilistic relaxation [Publicación periódica]. - pages 749–764 : IEEE Trans. PAMI 8, 1995.

Corrales J.C. [et al.] Bematch: A platform for matchmaking service behavior models [Sección del libro] // Proceedings of EDBT. - Nantes, France : [s.n.], 2008.

Corrales J.C. Behavioral matchmaking for service retrieval [Book]. - Versailles, France : tesis presentada a la University of Versailles Saint-Quentin-en-Yvelines para optar al grado de Doctor of Philosophy in Sciences, 2008.

Corrales J.C., Grigori D. and Bouzeghoub M. BPEL Processes Matchmaking for Service Discovery [Book Section]. - 2006.

Cover Robin Web Services Flow Language (WSFL) [Online]. - 02 04, 2002. - 05 13, 2010. - http://xml.coverpages.org/wsfl.html.

Doulkeridis [et al.] A system architecture for context-aware service discovery [Publicación periódica] // Electr. Notes Theor. Comput. Sci. 146. - 2006.

Doulkeridis C. and Vazirgiannis M. Querying and Updating a Context-Aware Service Directory in Mobile Environments [Book Section]. - [s.l.] : In Proc. 4th VLDB Workshop on Technologies on E-Services (TES'03), 2003.

ebXML Electronic Business eXtensible Markup Language [Online]. - 2006. - 05 13, 2010. - http://www.ebxml.org/.

El-Sayed Abdur-Rahman and Black J Semantic-based context-aware service discovery in pervasive-computing environments [Book Section]. - 2006.

Fajardo Ricardo y Miramá Víctor Protocolo para el Descubrimiento e Interacción de Servicios Ubicuos en una Ambiente Móvil [Libro]. - Popayán : Trabajo de Grado. Universidad del Cauca., 2008.

Fiebig T y Moerkotte g Algebraic XML construction and its optimization in Natix [Conferencia]. - 2002.

Georgantas Nikolaos [et al.] Semantic-aware Services for the Mobile Computing Environment [Sección del libro] // Architecting Dependable Systems III. - [s.l.] : Springer Verlag, 2005.

Gerbe O., Keller R. and Mineau G. Conceptual graphs for representing business processes in corporate memories [Book Section]. - 1998.

Gimeno JM. Computación Ubicua. // Capítulo 1. - [s.l.] : Revista La Flecha, 2004. - Vol. http://www.laflecha.net/articulos/ciencia/computacionubicua/.

Grigori D. and Bouzeghoub M. Service Retrieval Based on Behavioral Specification [Book Section]. - [s.l.] : SCC 2005, IEEE International Conference on Services Computing, Florida, 2005.

Guerrero Esteban, Corrales Juan Carlos y Ruggia Raul Selección de Servicios basado en Metamodelos del Perfil, Contexto y QoS [Conferencia] // EATIS. - 2010.

Hermida V., Caicedo, O., Corrales, J.C., Grigori, D. & Bouzeghoub, M. Service Composition Platform for Ubiquitous Environments Based on Service and Context Matchmaking [Conference]. - Bucaramanga, Colombia : Cuarto Congreso Colombiano de Computación 4CCC, Bucaramanga, 2009b.

Hermida Victor A. Arquitectura de Referencia para la Composición de Servicios en Ambientes de Computación Ubicua [Book]. - Popayán : Universidad del Cauca, 2009a.

Hermida Víctor Alberto [et al.] Plataforma para Descubrimiento de Servicios en Ambientes Ubicuos [Publicación periódica]. - [s.l.] : Revista Facultad de ingeniería Universidad de Antioquia, 2010.

Hopcroft R., Motwani J. E.and and Ullman J. D. Introduction to automata theory, languages, and computation [Book Section]. - 2001.

Jacobson I., Booch G. and Rumbaugh. J. The unified software development process. [Book Section]. - [s.l.] : In Addison Wesley, 1998.

Joines S., Willenborg R. and Hygh K. Performance Analysis for Java Websites [Book Section]. - [s.l.] : Addison-Wesley, ISBN-13: 978-0201844542, 2002.

Kenichi Yamazaki Research Directions for Ubiquitous Services. Applications and the Internet [Book Section]. - 2004. - Vols. ISBN: 0-7695-2068-5.

Köning-Rics Ulrich Küster and Brigitta Dynamic Binding for BPEL Processes - a Lightweight Aprproach to Integrate Semantics into Web Services [Conference] // ICSQC 2006. - 2007. - pp. 116-127.

Lewis D. Representation and learning in information retrieval [Book Section] // Ph.D. Thesis. - University of Massachusetts : Department of Computer and Information Science, 1992.

Martin David, Ankolekar Anupriya y Burstein Mark The OWL Services Coalition: OWL-S: Semantic Markup for Web Services [En línea]. - 2004. - 13 de 05 de 2010. - http://www.daml.org/services/owl-s/1.1/.

Martinez Francisco Marco de Referencia para la construcción de Aplicaciones de Comercio Electrónico Móvil en Países en Vía de Desarrollo [Book Section]. - Universidad del Cauca. Popayán : Tesis de Maestría, 2008.

McCown Sean Databases Flex Their XML: IBM, Microsoft, Oracle, and Sybase Compete in Our Data Management Gymnastics. [Conference] // InfoWorld. - 2004.

Mendling J. and Ziemann J. Transformation of bpel processes to epcs. [Book Section]. - [s.l.] : In Proc. of the 4th GI Workshop on Event-Driven Process Chains (EPK2005), 2005.

Mendling J. y Ziemann J. Transformation of bpel processes to epcs [Conferencia] // Proc. of the 4th GI Workshop on Event-Driven Process Chains (EPK2005). - 2005.

Miller G. Wordnet: A lexical database for english. [Book Section] // Communications of the ACM. - [s.l.] : 38(11):39–41, 1995.

Milner R. A calculus of communicating systems [Book Section]. - Springer-Verlag NewYork, Inc : [s.n.], 1982.

Mongiello Marina y Castelluccia Daniela Modelling and verification of BPEL business processes [Conferencia] // ourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software. - Potsdam, Germany : [s.n.], 2006. - págs. 144-148.

Papazoglou M. P. Service-oriented computing [Journal] // Special section in Communications of the ACM . - [s.l.] : ACM Press, 2003.

Passani L. Wurlf [Online]. - 2007. - Octubre 7, 2009. - http://wurfl.sourceforge.net/.

Pautasso C. and Alonso G Visual composition of web services [Conference] // Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments. - 2003. - pp. 92–99.

Peterson J. L. Petri net theory and the modeling of systems [Book Section]. - [s.l.] : In Prentice-Hall, 1981.

R.Wilson and Hancock E. Graph matching by discrete relaxation. Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies and Hybrid Systems [Book Section]. - pages 165–176 : [s.n.], 1994.

Rudi Belotti Corsin Decurtins , Michael Grossniklaus , Moira C. Norrie and Alexios Palinginis Modelling Context for Information Environments [Conference] // WorkShop on Ubiquitous Mobile Information and Collaboration Systems. - Riga, Latvia : Springer Berlin / Heidelberg, 2004. - pp. 43-56.

Sellami Mohamed, Tata Samir y Defude Bruno Service Discovery in Ubiquitous Environments: Approaches and Requirement for Context-Awareness [Conferencia] // In Advances in Semantics for Web services Workshop. - Milan, Italy : BPM Workshops, 2009.

Shen Z. y Su J. Web services discovery based on behavior signatures [Sección del libro] // Proceedings of IEEE SCC. - Orlando, Florida : [s.n.], 2005.

Sivashanmugam [et al.] Discovery of web services in a federated registry environment [Journal] // Proceedings of the IEEE International Conference on Web Services . - Washington, DC, USA : IEEE Computer Society , 2004.

Spendolini Michael J. Benchmarking [Book]. - [s.l.] : Grupo Editorial Norma, 1994.

Steller Luke y Krishnaswamy Shonali Efficient Mobile Reasoning for Pervasive Discovery [Conferencia] // Proceedings of the 2009 ACM symposium on Applied Computing (SAC 2009: 1247-1251).. - [s.l.] : ACM, 2009.

Steller Luke, Krishnaswamy Shonali y Newmarch Jan Discovering Relevant Services in Pervasive Environments Using Semantics and Context [Conferencia] // Proceedings of ACM (IWUC 2006: 3-12).. - [s.l.] : ACM, 2006.

Tocarruncho Tocarruncho Sandra Patricia, Aponte Novoa Fredy Andrés y Tocarruncho Tocarruncho Arturo EXTRACCIÓN DE PERFILES BASADA EN AGRUPAMIENTO GENETICO PARA RECOMENDACIÓN DE CONTENIDO [Conferencia] // Conferência IADIS Ibero-Americana WWW/Internet 2007. - 2007.

Vanhatalo J., Koehler J. y Leymann F. Repository for business processes and arbitrary associated metadata [Sección del libro]. - 2006.

vaquita J., Koehler J. and Leymann F. Repository for business processes and arbitrary associated metadata [Book Section]. - 2006..

W3C (WS-CDL), Web Services Choreography Description Language Version 1.0 [En línea]. - 17 de 12 de 2004. - 13 de 15 de 2010. - http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/.

W3C W3C. "Guía Breve de Servicios Web" [Online]. - 05 06, 2010. - 05 13, 2010. - http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

Weiser Mark The Computer for the 21st Century [Book Section]. - 1991.

Weiser Mark Ubiquitous Computing [Book Section]. - 1997.

Weiser Ubiquitous Computing [Online]. - 03 17, 96. - 05 2010, 13. - http://sandbox.xerox.com/ubicomp/.

WfMC The Workflow Management Coalition [Online]. - 01 17, 2008. - http://www.wfmc.org.

Wombacher A. [et al.] Matchmaking for business processes based on choreographies [Sección del libro] // Proceedings of EEE. - Washington DC : [s.n.], 2004.

Wombacher A., Mahleko B. and Fankhauser P. A grammar-based index for matching business processes [Book Section] // Proceedings of ICWS. - Washington DC : [s.n.], 2005.

Xu [et al.] Semantic web services discovery in p2p environment [Conferencia] // ICPPW'07. - 2007.

YAWL YAWL Yet Another Workflow Language [Online]. - 2010. - 10 20, 2008. - http://www.yawl-system.com/.

Zhang Mi y Hurley Neil Novel Item Recommendation by User Profile Partitioning [Conferencia] // 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. - Milan, Italy : [s.n.], 2009.

Zhu H. and Zhou M. Role-Based Multi-Agent Systems. Personalized Information Retrieval and Access: Concepts, Methods, and Practices [Journal] // Information Science Reference. - 2008. - pp. 254-285..

ANEXO A

ARTEFACTOS OBTENIDOS EN LA PLANEACIÓN, CONSTRUCCIÓN Y ELABORACIÓN DE LA PLATAFORMA U-ServiceMatch

A.1 Fase de Planeación y Elaboración de U-ServiceMatch

Esta sección presenta los resultados del proceso de ingeniería llevado a cabo para el desarrollo de la arquitectura que soporta la plataforma propuesta.

Para el desarrollo de los componentes software que conforma la plataforma propuesta, se utilizó la metodología sugerida en el Modelo de Construcción de Soluciones (Serrano, 05). La cual plantea el desarrollo de un prototipo de valoración considerando las siguientes fases: Establecimiento de responsabilidades, Descripción de la solución, Definición de la Arquitectura, Implementación y Puesta a Punto, y Evaluación. Para éste proyecto la etapa de Implementación y Puesta a Punto se desarrolló en iteraciones incrementales incluyendo actividades de Verificación y Validación al final de cada iteración.

Se plantearon las siguientes iteraciones:

Iteración 1: Implementación del prototipo de Emparejamiento de Actividades BPEL, Integración del Repositorio de Servicios con el módulo de Emparejamiento (ServiceMatch), Verificación y Validación.

Iteración 2: Definición del metadato que describe los requerimientos de contexto del servicio, Implementación del mecanismo de consulta de los metadatos de contexto. Adaptación del prototipo ServiceMatch al contexto, Implementación del Repositorio de Dispositivos, Integración del prototipo ServiceMatch con el Repositorio de Capacidades de Dispositivos (SeMatch-Context), Verificación y Validación.

Iteración 3: Implementación del Repositorio de Usuarios, Integración del prototipo SeMatch-Context con el Repositorio de Usuarios (U-ServiceMatch), Verificación y Validación.

5.3.9. Casos de Uso

En esta sección se presenta el diagrama de casos de uso de la plataforma U-ServiceMatch desarrollada en el presente trabajo de grado. Los casos de uso se identificaron a partir de las funcionalidades que ofrece el sistema.

5.3.9.1. Diagrama de Casos de Uso de la Plataforma U-ServiceMatch

La Figura A1 presenta el diagrama de casos de uso general para la plataforma U-ServiceMatch, donde se pueden apreciar los tipos de usuarios que interactúan con la aplicación y las funcionalidades del sistema.

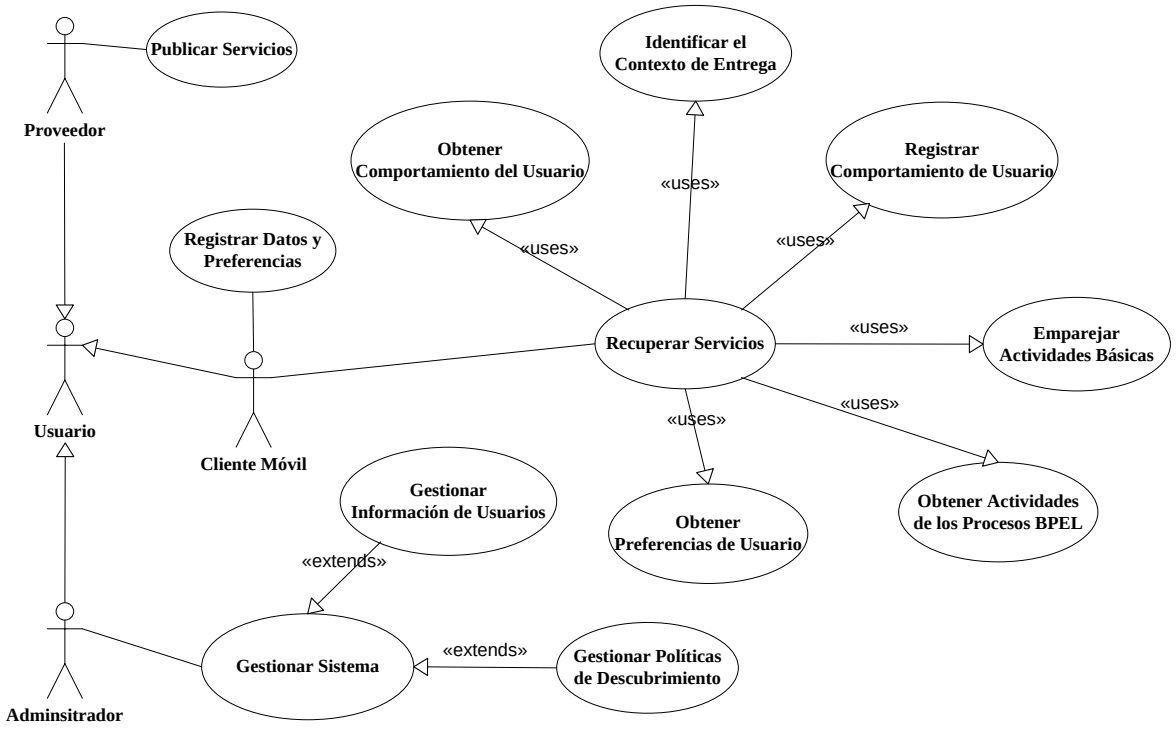


Figura A 1. Diagrama de Casos de Uso de la Plataforma U-ServiceMatch.

A.1.1.2 Casos de uso en formato expandido

En esta sección se presenta los casos de uso en formato expandido para la Plataforma U-ServiceMatch.

Caso de uso:	**REGISTRAR DATOS Y PREFERENCIAS**
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Ingresar al sistema.
Resumen:	Permite registrar información personal y preferencias de usuario, para posteriormente crear su perfil dentro del sistema.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Éste caso de uso inicia cuando un usuario quiere ingresar al sistema por primera vez. 3. El usuario ingresa sus datos y preferencias personales para realizar su ingreso y posteriormente crear el *Perfil de Usuario*. 4. El usuario ingresa a la plataforma.	2. El sistema presenta la interfaz de Registro de la Plataforma U-ServiceMatch al nuevo usuario.

Tabla A 1. Caso de uso en formato expandido Registrar Datos y Preferencias.

Caso de uso:	PUBLICAR SERVICIOS
Actores:	Usuario – Proveedor (Iniciador).
Propósito:	Publicar Servicios.
Resumen:	Permite publicar los documentos BPEL y los requerimientos del contexto asociado a dicho proceso en el Repositorio de Servicios.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Éste caso de uso inicia cuando un proveedor quiere publicar sus servicios en el *Repositorio de Servicios* del sistema. 3. El usuario valida el formato del documento BPEL según las políticas del sistema y además, selecciona los requerimiento de contexto del servicio a publicar. 4. El proveedor publica sus servicios en la plataforma.	2. El sistema presenta al proveedor la interfaz de Publicación de Servicios.

Tabla A 2. Caso de uso en formato expandido Publicar Servicios.

Caso de uso:	RECUPERAR SERVICIOS
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Recuperar Servicios.
Resumen:	Éste caso de uso permite a un cliente móvil consultar y recuperar los servicios más relevantes que se ajusten a su petición, mediante una fase emparejamiento a nivel atómico de actividades BPEL, considerando tanto las restricciones del contexto, como las preferencias del usuario..
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
2. Éste caso de uso inicia cuando un cliente móvil desea usar la funcionalidad: *Buscar Servicios* ofrecida por la Plataforma U-ServiceMatch. 4. El usuario solicita a la plataforma encontrar los servicios más adecuados a los requerimientos de su petición. 7. El usuario selecciona los servicios que desee y satisfagan sus necesidades.	3. El sistema presenta al usuario la interfaz de consulta de servicios. 5. El sistema inicia el proceso de recuperación de los servicios más pertinentes a la consulta del usuario. 6. El sistema retorna un ranking de los servicios más relevantes encontrados que pueden ser consumidos por el cliente móvil y un ranking de servicios sugeridos, los cuales no pueden ser consumidos desde el terminal móvil que hizo la consulta.

Tabla A 3. Caso de uso en formato expandido Recuperar Servicios.

Caso de uso:	OBTENER COMPORTAMIENTO DEL USUARIO
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Obtener el comportamiento del Usuario
Resumen:	Éste es caso de uso permite a la plataforma obtener información del comportamiento del usuario capturada dinámicamente, tal como servicios consumidos y patrones de uso del sistema.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
3. Éste caso de uso inicia cuando un cliente móvil realiza las siguientes acciones: Consulta servicios, seleccionar servicios consultados y consumir servicios. 5. El usuario solicita a la plataforma encontrar los servicios más adecuados a los requerimientos de su petición. 8. El usuario selecciona los servicios que desee y satisfagan sus necesidades.	4. El sistema presenta al usuario la interfaz de consulta de servicios. 7. El sistema inicia el proceso de recuperación de los servicios más pertinentes a la consulta del usuario. 8. El sistema retorna un ranking de los servicios más relevantes encontrados que pueden ser consumidos por el cliente móvil y un ranking de servicios sugeridos, los cuales no pueden ser consumidos desde el terminal móvil que hizo la consulta.

Tabla A 4. Caso de uso en formato expandido Obtener Comportamiento del Usuario.

Caso de uso:	OBTENER PREFERENCIAS DEL USUARIO
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Obtener las preferencias del Usuario
Resumen:	Éste caso de uso permite obtener el perfil de usuario para la fase de descubrimiento, ya que es conveniente considerar las preferencias de usuario durante éste proceso.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario solicita a la plataforma encontrar los servicios más adecuados a los requerimientos de su petición.	2. El sistema obtiene el perfil del usuario y lo compara con perfiles similares para ofrecer una mayor cantidad de posibles servicios relevantes a la solicitud (historial de servicios consultados y consumidos), con el fin de agilizar el proceso de recuperación.

Tabla A 5. Caso de uso en formato expandido Obtener Preferencias del Usuario.

Caso de uso:	OBTENER ACTIVIDADES DE LOS PROCESOS BPEL
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Obtener las actividades básicas de los procesos BPEL.
Resumen:	Éste caso de uso permite obtener, filtrar y organizar las actividades básicas, tanto aquellas que conforman el requisito del cliente, como las contenidas en los procesos BPEL del *Repositorio de Servicios*. Ya que el proceso de descubrimiento se apoya de una fase de emparejamiento a nivel atómico.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Éste caso de uso inicia cuando el usuario solicita a la plataforma encontrar los servicios más adecuados a los requerimientos de su petición.	2. El sistema obtiene el perfil del usuario y lo compara con perfiles similares para ofrecer una mayor cantidad de posibles servicios relevantes a la solicitud (historial de servicios consultados y consumidos), con el fin de agilizar el proceso de recuperación.

Tabla A 6. Caso de uso en formato expandido Obtener Actividades de los Procesos BPEL.

Caso de uso:	REGISTRAR COMPORTAMIENTO DE USUARIO
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Obtener y Registrar el comportamiento del Usuario
Resumen:	Una vez el proceso de recuperación de servicios ha sido ejecutado, el sistema registra tanto los servicios consumidos, como los que simplemente han sido consultados. Éste caso de uso permite registrar dicho comportamiento en el *Repositorio de Usuarios* y actualizar el perfil de los usuarios.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. El usuario selecciona (del ranking de servicios encontrados que pueden ser consumidos y/o del ranking de servicios sugeridos, los cuales no pueden ser consumidos desde el terminal móvil que hizo la consulta) los servicios más pertinentes que satisfagan sus requerimientos.	2. El sistema registra el comportamiento del usuario, es decir: los servicios consultados y consumidos, y los dispositivos usados para ello.

Tabla A 7. Caso de uso en formato expandido Registrar Comportamiento de Usuario.

Caso de uso:	EMPAREJAR ACTIVIDADES BÁSICAS
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Comparar dos actividades para determinar su similitud
Resumen:	Éste caso de uso permite comparar dos nodos, que representan actividades básicas de BPEL (*receive, invoke* y *reply*), y calcular la distancia semántica entre ellos.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Éste caso de uso inicia cuando el usuario solicita a la plataforma encontrar servicios que se adecuen los requerimientos de su petición.	2. Verifica que los nodos a comparar tengan el mismo tipo de actividad (*receive, invoke* y *reply*). 3. El sistema toma dos nodos del mismo tipo de actividad y los compara mediante el uso de algoritmo que permiten determinar la distancia semántica entre ellos. 4. Una vez comparados los nodos *Query* (consulta del usuario) contra los nodos *Target* (almacenados en el repositorio). Organiza los nodos más pertinentes según su valor de similitud. Entregando un ranking de los servicios más adecuados.

Tabla A 8. Caso de uso en formato expandido Emparejar Actividades Básicas.

Caso de uso:	IDENTIFICAR EL CONTEXTO DE ENTREGA
Actores:	Usuario – Cliente Móvil (Iniciador).
Propósito:	Comparar dos actividades para determinar su similitud
Resumen:	Éste caso de uso permite identificar las restricciones del solicitante y obtener los requerimientos del servicio. Información relevante para el proceso de descubrimiento de servicios en entornos tan dinámicos como los ambientes ubicuos
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
1. Éste caso de uso inicia cuando el usuario ingresa y solicita a la plataforma encontrar servicios.	2. El sistema identifica las cabeceras HTTP y recolecta la información relacionada con el dispositivo dentro de WURFL y UAProf. 3. Obtiene las capacidades del cliente móvil y crea el contexto de entrega. 4. EL sistema obtiene las restricciones del contexto en el cual el servicio recuperado puede ser consumido.

Tabla A 9. Caso de uso en formato expandido Identificar el Contexto de Entrega.

Caso de uso:	GESTIONAR SISTEMA
Actores:	Usuario – Administrador (Iniciador).
Propósito:	Administrar las funcionalidades del Sistema
Resumen:	Éste caso de uso le permite al usuario administrador ciertos privilegios a la hora de interactuar con la plataforma, tales como: gestionar toda la información de usuarios y las políticas de descubrimiento del sistema.
Tipo:	Primario y extendido.
Curso normal de eventos	
Acción de los Actores	**Respuesta del sistema**
5. Éste caso de uso inicia cuando el usuario administrador ingresa a la plataforma para gestionar la información de los usuarios registrados y configurar las políticas de descubrimiento.	6. El sistema despliega la interfaz de administración de la plataforma U-ServiceMatch. 7. El sistema permite al administrador gestionar toda la información relacionada con los usuarios de la plataforma. 8. El sistema brinda al administrador la facilidad de configurar las políticas de descubrimiento y emparejamiento de servicios de la plataforma.

Tabla A 10. Caso de uso en formato expandido Gestionar Sistema.

A.2 Fase de Construcción

5.3.10. Diagramas de clases del Sistema

En esta sección se expone el diagramas de clases (sin los atributos y métodos que las conforman, con el objetivo de hacer legible su presentación en éste documento) de las aplicaciones software definidas que componen la plataforma U-ServiceMatch: *ServiceMatch*, *SeMatch-Contex* y *Repositorio de Usuarios.* Las descripciones de las clases de cada diagrama son también presentadas en esta sección.

A.2.1.1 Diagrama de Clases del Analizador BPEL - Grafos

La Figura A2 solo esquematiza el fin último de este proyecto, pero internamente cuenta con otros procesos como el de lectura de documentos BPEL o generación de un meta-modelo de grafos. Para complementar esto, su funcionamiento general se describe a continuación.

Inicialmente la clase *BpelReader* carga el archivo (.bpel) y escoge un lector especializado para cada actividad. Estos lectores no son más que clases, las cuales implementan la interfaz *IActivityReader*. Esta interfaz se configura manipulando un archivo XML el cual especifica la lectura de cada actividad gracias a las etiquetas <*activityReader*> las cuales tienen dos atributos, el primero corresponde al nombre de la actividad que va a ser procesada y el segundo a la clase Java que realiza la lectura. Seguido, se almacena todo lo concerniente a una representación en objetos del documento BPEL leído, para cada parte del archivo de entrada se extraen sus componentes y se crean objetos que representan fielmente al proceso de negocio.

La parte más importante del analizador es la clase *BpelTransform*, responsable de transformar el metamodelo de BPEL en un objeto de tipo *ProcessGraph.* Éste es el punto de entrada o el punto de inicio para la transformación de un proceso BPEL en un Grafo. La interfaz *IActivityTransform*, debe ser implementada por cualquier clase que quiera transformar una actividad en particular. En conclusión, la clase *BpelTransform*, tiene las estrategias de transformación de cada actividad BPEL. Otra clase importante del paquete en cuestión y en cuanto a estrategias se refiere, es la *AbstractActivityTransform*, la cual contiene todas las estrategias de transformación comunes a todas las actividades.

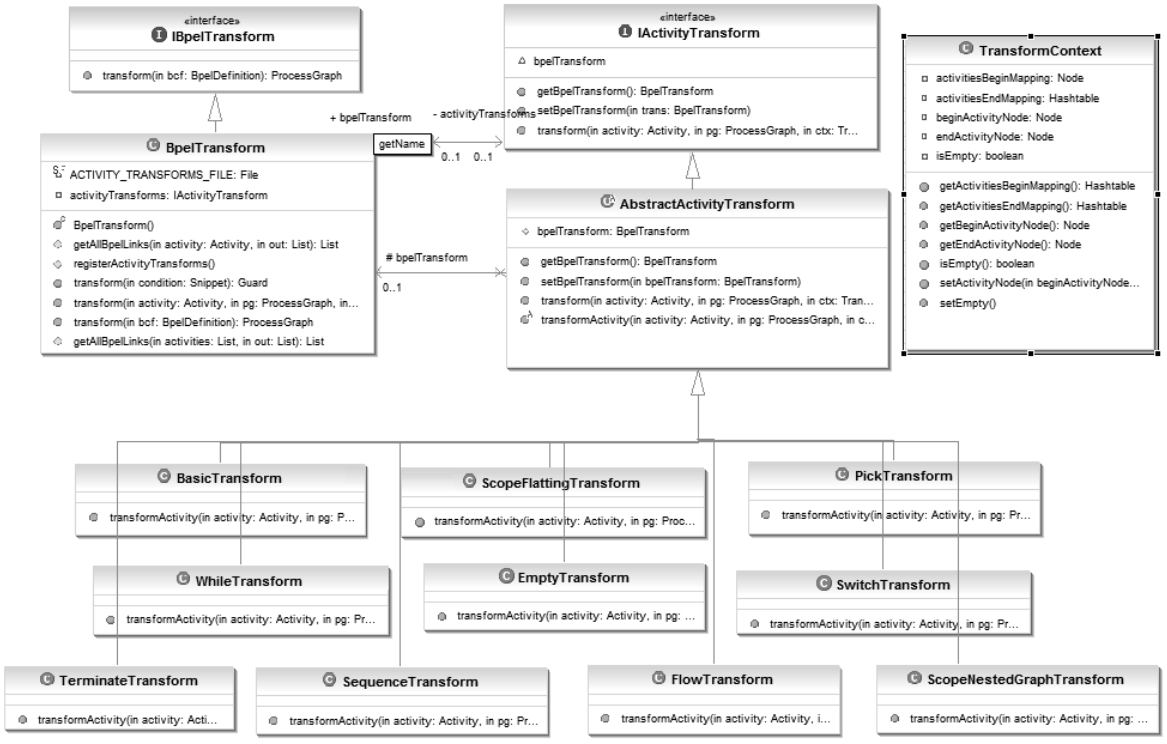


Figura A 2. Diagrama de Clases para la Transformación de un documento BPEL a Grafos

A.2.1.2 Diagrama de Clases de la aplicación ServiceMatch

La Figura A3 presenta el diagrama de clases de la aplicación ServiceMatch, las cuales han sido agrupadas según la funcionalidad que implementen: Gestión del Repositorio de Servicios, Emparejamiento de Servicios, Manejo de Grafos y Presentación.

A continuación se presenta una descripción detallada de las clases que conforman la aplicación ServiceMatch.

- **Gestión del Repositorio de Servicios:** a continuación se describen las clases que implementan funcionalidades que permiten manipular el *Repositorio de Servicios.* Estas clases son:

- *BPELRepositoryManager*: ésta clase contiene la lógica que permite gestionar, tanto las consultas como la colección de extensiones EMF (*Eclipse Modeling Framework*) del Repositorio de Procesos BPEL.

- *DefaultEMFExtensionCollection*: una de las más importantes características del Repositorio de Procesos de Negocio es la posibilidad de registrar múltiples modelos EMF, para soportar el almacenamiento de diferentes tipos de archivos, extendiendo así su capacidad. Ésta clase permite registrar los modelos para BPEL, WSDL, XSD (por defecto soportados dentro del repositorio) y otros.

- *KentOCLQueryEngineImpl*: ésta clase implementa el motor de consulta principal de la versión pública del *Repositorio de Servicios* desarrollado por la Universidad de Kent. Es un motor de consulta OCL de código abierto que implementa el estándar OCL 2.0. Para éste prototipo, el motor OCL de Kent obtiene las URI de los archivos BPEL almacenados en el repositorio de procesos de negocio.

- *BPELRepository*: ésta clase permite inicializar el Repositorio de Procesos, registrando tanto objetos EMF como los motores OCL de consulta.

- *BpelReader*: ésta clase carga los archivos *.bpel* encontrados en el repositorio, los lee, mediantes lectores especializados para cada actividad, y almacena toda la información necesaria para el proceso de transformación. Para cada parte del archivo de entrada se extraen sus componentes y se crean objetos que representan finalmente el procesos de negocio.

- *ProcessGraph*: ésta clase permite almacenar en un metamodelo de objetos la estructura resultante de la serialización del documento BPEL, representando finalmente un grafo mediante arcos, funciones, conectores y nodos *start/end.* Las funciones son los nodos del grafo que representan acciones específicas. Los nodos *start/end* representan el nodo inicial y el nodo final del grafo. Los conectores son nodos que hacen posible las conexiones entre elementos al igual que pueden llevar a cabo diversas conexiones en el proceso.

- *BpelTransform*: ésta clase implementa las estrategias de transformación de cada actividad BPEL, siendo la responsable de transformar el metadato de BPEL en un Objeto Java de tipo *ProcessGraph*. Ésta clase contiene la lógica para implementar el *Analizador BPEL a Grafos*, el cual es un proyecto desarrollado en Java, que define el diagrama de clases presentado en la Figura A2 para realizar la transformación de un documento BPEL a grafos.

- *ControlBPELTransform*: ésta clase implementa la lógica de control que permite almacenar, encontrar, usar y definir archivos basados en el esquema estándar XML. Además, permite transformar archivos BPEL en su equivalente en Grafos.

❖ **Manejo de Grafos:** a continuación se describen las clases que implementan funcionalidades que permiten operar sobre los Grafos abstraídos de los procesos encontrados en el *Repositorio de Servicios.* Estas clases son:

- *ActivityRepository*: ésta clase contiene la lógica para obtener las funciones/nodos de los Grafos de procesos, para organizarlos y almacenarlos por tipo de actividad (*invoke, reply, receive*).

- *Activity*: ésta clase representa la actividad/nodo/función de un Grafo. La cual puede ser: *FunctionInvoke*, *FunctionReceive*, *FunctionReply*. Cada función posee cuatro atributos: *Tipos de Actividad, Nombre de la operación PartnerLink* y *PortType.*

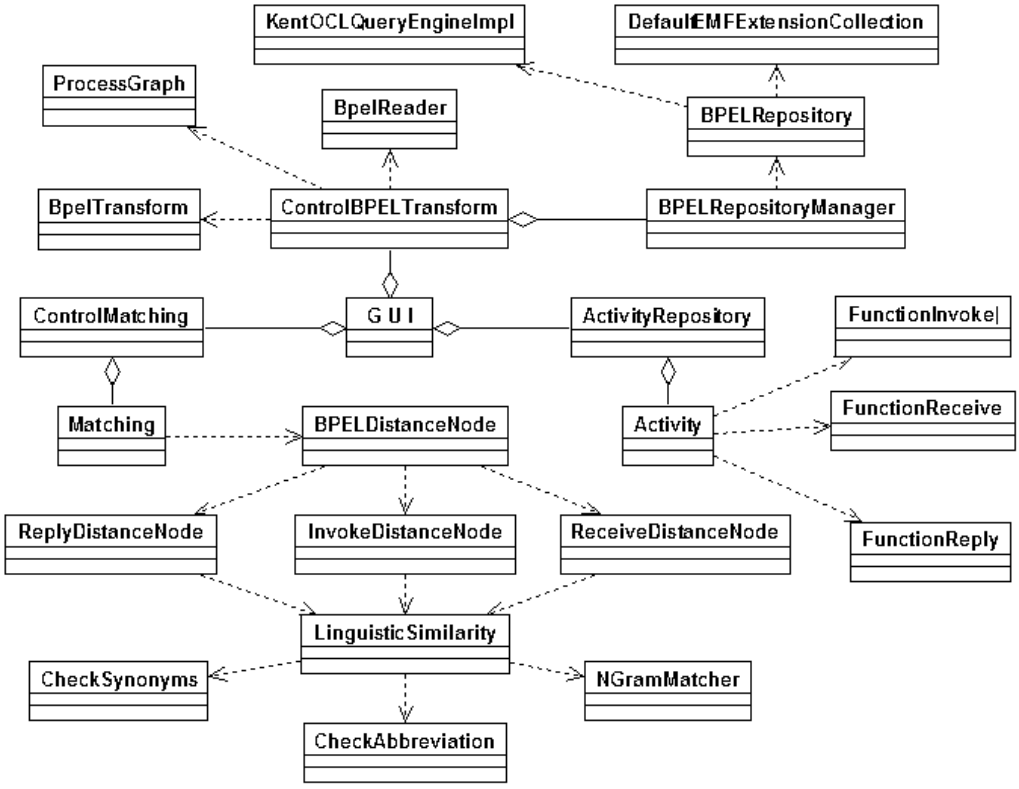


Figura A 3. Diagrama de Clases aplicación ServiceMatch

❖ **Emparejamiento de Servicios:** a continuación se describen las clases que implementan funcionalidades que soportan el proceso de emparejamiento de servicios. Estas clases son:

- *Matching*: está clase toma como entradas dos nodos, que representan actividades básicas de BPEL (*receive, invoke, reply*), y calcula la distancia semántica entre ellos. Para esto utiliza las clases: *InvokeDistanceNode*, *ReceiveDistanceNode y ReplyDistanceNode*; quienes a su vez usan la clase *LinguisticSimilarity.*

- *BPELDistanceNode*: ésta clase implementa la lógica que permite verificar que los nodos a ser comparados tengan el mismo tipo de actividad.

- *InvokeDistanceNode*: ésta clase permite obtener la distancia semántica entre dos nodos tipo *Invoke*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *ReceiveDistanceNode*: ésta clase permite obtener la distancia semántica entre dos nodos tipo *Receive*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *ReplyDistanceNode*: ésta clase permite obtener la distancia semántica entre dos nodos tipo *Reply*, para lo cual se fijan los pesos *Wop, Wpl* y *Wpt* que indican la contribución de la similitud de *Operación*, *PartnerLink* y *PortType*, respectivamente, a la similitud de las actividades.

- *LinguisticSimilarity*: calcula la similitud lingüística entre dos etiquetas basándose en sus nombres. Para obtener esta medida se utilizan los algoritmos *Ngram, Check synonym* y *Check abbreviation*, implementados por las clases *CheckAbbreviation*, *CheckSynonyms* y *NGramMatcher* respectivamente.

- *CheckAbbreviation*: ésta clase implementa la lógica que le permite estimar la similitud de dos cadenas usando un diccionario de abreviaciones adecuado al dominio de aplicación.

- *CheckSynonyms*: ésta clase implementa la lógica que le permite estimar la similitud de dos cadenas usando el diccionario lingüístico Wordnet, para identificar sinónimos.

- *NGramMatcher*: ésta clase implementa la lógica que le permite estimar la similitud de dos cadenas de acuerdo al número común de *qgramas* entre las etiquetas.

❖ **Presentación:** a continuación se describen las clases que implementan funcionalidades de representación visual del prototipo *ServiceMatch.* Estas clases son:

- *GUI*: ésta clase implementa la lógica de presentación para las interfaces gráficas del prototipo *ServiceMatch*.

- *ControlMatching*: ésta clase implementa el control de los datos entre los procesos de tratamiento y visualización de la información.

A.2.1.3 Diagrama de Clases de la aplicación SeMatch-Context

La Figura A5 presenta el diagrama de clases de la aplicación SeMatch-Context, las cuales son descritas a continuación:

- *ContextQueryEngine*: ésta clase es utilizada por *ControlBPELTransform* de la aplicación *ServiceMatch* expuesta, la cual implementa un mecanismo de consulta del valor correspondiente al atributo de contexto (*Tipo de Acceso*) de los servicios publicados en el repositorio.

- *ServiceMatch*: representa el *.JAR* (*Java ARchive*) del prototipo ServiceMatch definido. Los *.jar* son un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Dada la lógica implementada en el *ServiceMatch*, permite determinar la similitud

entre un nodo *Query* y una colección de nodos *Target* (1:N), y entregar una lista ordenada de las actividades más similares a un nodo de consulta.

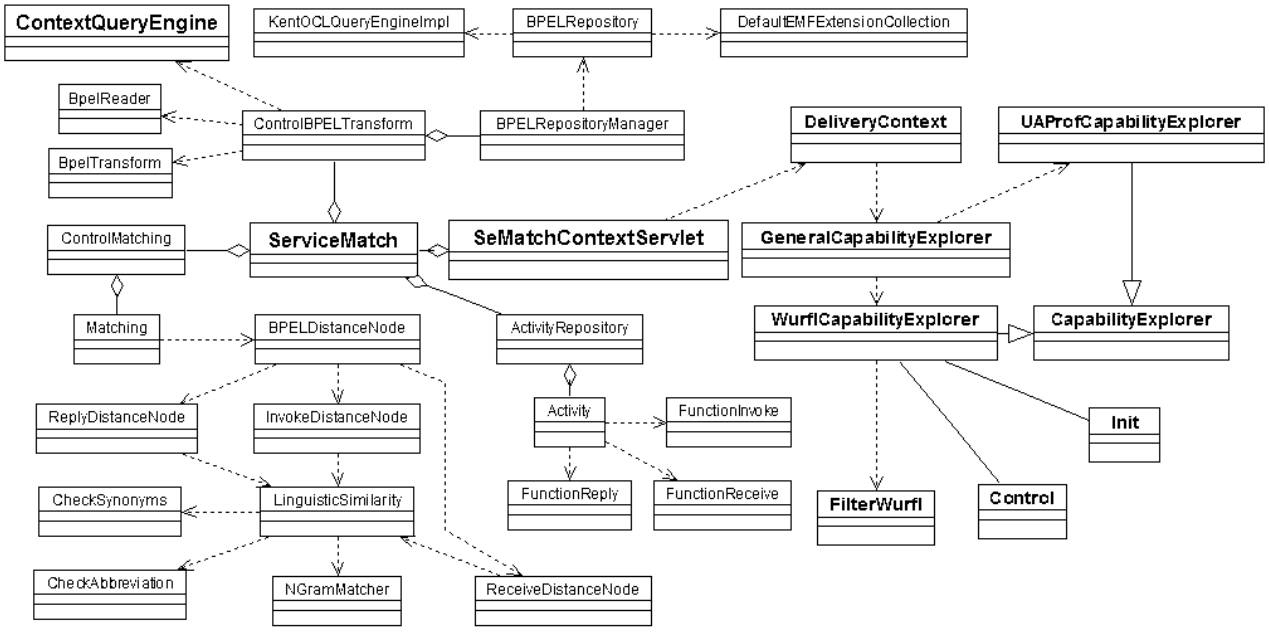


Figura A 4. Diagrama de Clases aplicación SeMatch-Match

- *SeMatchContextServlet*: su función es atender las peticiones provenientes de un cliente móvil atreves de su navegador Web. Ésta clase implementa la lógica que le permite al prototipo *SeMatch-Context* obtener el contexto de entrega de los clientes que acceden a la aplicación. Además, controla el proceso de recuperación de servicios, considerando tanto el requerimiento del contexto de los servicios, como las restricciones de contexto del usuario.

- *DeliveryContext*: ésta clase se encarga de obtener el contexto de entrega, utilizando tres fuentes de información: Cabeceras HTTP, UAProf y WURFL.

- *GeneralCapabilityExplorer*: ésta clase permite explorar las capacidades de un dispositivo móvil, obteniendo la información de la biblioteca WURFL o de los repositorios UAProf.

- *UAProfCapabilityExplorer*: ésta clase se encarga de explorar las capacidades de un móvil obteniendo la información en los repositorios UAProf.

- *WurflCapabilityExplorer*: ésta clase se encarga de explorar las capacidades de un móvil obteniendo la información en la biblioteca WURFL.

- *CapabilityExplorer*: es la Interfaz que define la estructura de los exploradores de capacidades para dispositivos móviles.

- *Init*: clase encargada de cargar en RAM el archivo "*wurfl.xml*" que contiene la información de la biblioteca WURFL.

- *Control*: clase encargada de controlar la obtención de información en las bibliotecas WURFL.

- *FilterWurfl*: ésta clase permite buscar las capacidades de un dispositivo en la biblioteca WURFL mediante la cabecera HTTP, "*User-Agent*".

A.2.1.4. Diagrama de Clases del Repositorio de Usuarios

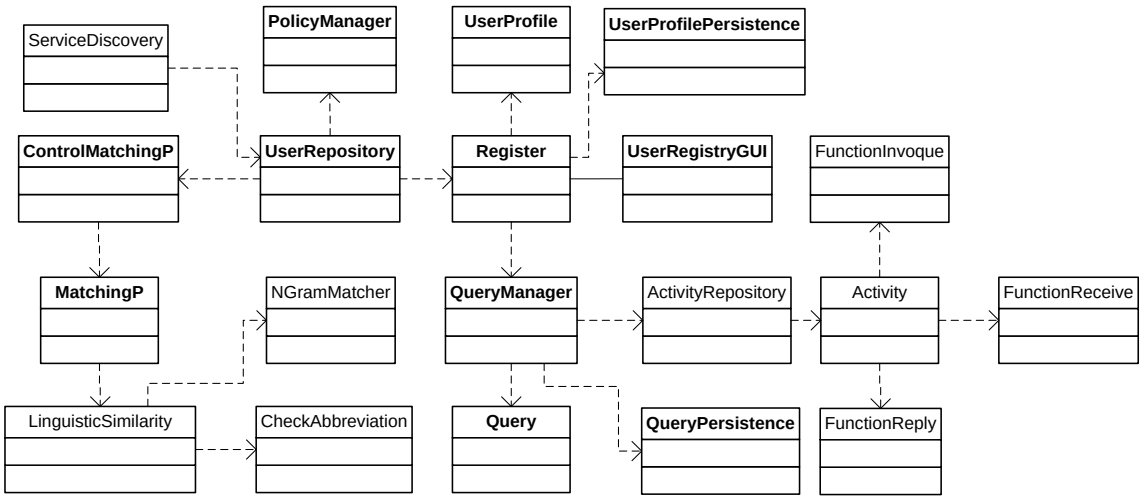


Figura A 5. Diagrama de Clases Repositorio de Usuarios

A.2.2 Diagramas de paquetes del Sistema

Los diagramas presentados en esta sección exponen la vista lógica de las aplicaciones software que componen el sistema. Dichos diagramas se organizan en paquetes, subsistemas y capas (*Aplicación, mediación, y Almacenamiento y Servicios*) (Jacobson, y otros, 1998). Además, se muestra la interacción existente entre capas, así como los paquetes más relevantes que las componen.

A.2.2.1 Diagrama de Paquetes para la aplicación ServiceMatch

La Figura A6 presenta la vista lógica de la aplicación ServiceMatch. A continuación se describen las capas y su interacción, así como los paquetes más relevantes que las componen.

- **Capa de aplicación:** contiene los paquetes que implementas las funcionalidades del prototipo, estos son:

  - *Analizador BPEL-Grafos*: éste paquete permite registrar funciones que transforman meta-modelos de servicios en su equivalente en grafos. Para el prototipo ServiceMatch se ha implementado las funciones de BPEL. Sin embargo, es posible

registrar otro tipo de funciones y extender su aplicabilidad a otros modelos como WSDL o WS-CDL.

- *Clasificador de Actividades*: éste paquete contiene las clases que implementan la lógica para obtener y clasificar los nodos, tanto del grafo de entrada, como de los abstraídos del *Repositorio de Servicios*. Para ellos usa el *Almacenador de Nodos por tipo de Actividad* y el *Filtro de Actividades*.

- *Almacenador de Nodos por tipo de Actividad*: éste paquete contiene las clases que permiten obtener y clasificar los nodos de un grafo por tipo de actividad (*receive, invoke* y *reply*). Paralelamente, almacena los nodos obtenidos, verificando, con ayuda del *Filtro de Actividades,* que no estén repetidos.

- *Filtro de Actividades*: éste paquete contiene las clases que implementan un algoritmo que filtrar las actividades que ya han sido almacenadas.

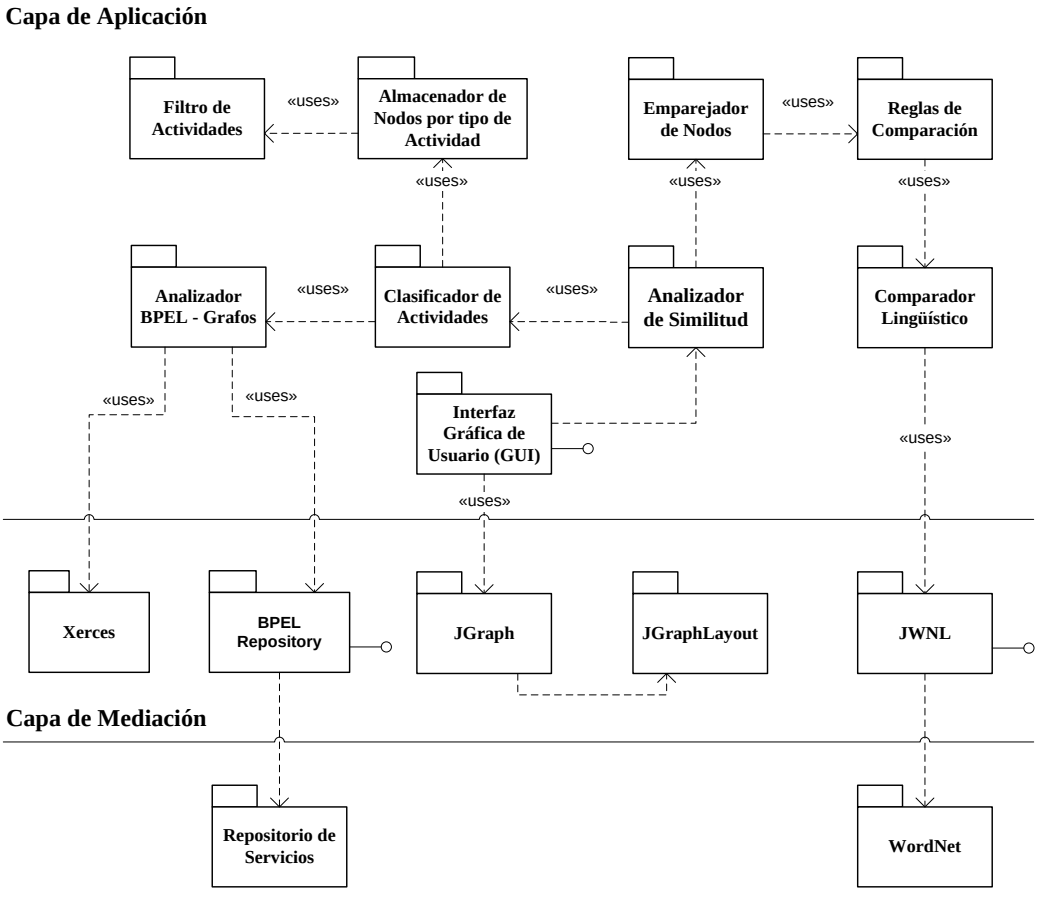


Figura A 6. Vista lógica de la Aplicación ServiceMatch

- *Analizador de Similitud*: contiene las clases que implementan la lógica para determinar la similitud entre un nodo *Query* y una colección de nodos *Target* (1:N), y

así entregar una lista ordenada de las actividades más similares al nodo de consulta. Éste paquete usa el *Emparejador de Nodos, Reglas de Comparación* y el *Comparador Lingüístico* para calcular dicha coincidencia.

- *Emparejador de Nodos*: implementa la lógica que permite determinar la similitud de dos nodos *Query* y *Target*, basado en la distancia 16emántica (proporcionada por los módulos *Reglas de Comparación* y *Comparador Lingüístico*) entre ellos.

- *Comparador Lingüístico*: contiene las clases implementan la lógica para calcular la similitud entre dos cadenas. Por ejemplo, para el ServiceMatch se ha utilizado los algoritmos *Ngram*, *Sinonym* y *Abbreviation*. Sin embargo, otros algoritmos para tal fin pueden ser registrados.

- *Interfaz Gráfica de Usuario (GUI)*: Con el fin de lograr una representación visual, este paquete contiene todas las clases que implementan las interfaces gráficas del prototipo ServiceMatch.

❖ **Capa de Mediación:** contiene todas las interfaces de programación de aplicaciones (APIs) utilizadas por el prototipo. La capa está compuesta por los siguientes paquetes:

- *Xerces*: es un Parser XML compuesto por una familia de paquetes software que permiten analizar y manipular archivos XML. La biblioteca implementa una serie de APIs estándar que incluye a DOM, SAX y SAX2.

- *BPEL Repository*: es una API de Java que permite manipular documentos BPEL (y otros archivos XML relacionados), encontrados en el *Repositorio de Servicios,* como objetos. Ocultando la serialización y deserialización al usuario. Las consultas son realizadas utilizando un lenguaje de consulta orientado a objetos, OCL.

- *JGraph:* es una librería que contiene todas las funcionalidades de visualización e interacción gráfica. Éste paquete permite la visualización de la interfaz gráfica de usuario (GUI).

- *JgraphLayout*: es una librería de diseño gráfico de alto rendimiento para Jgraph que automáticamente posiciona una gráfica, un diagrama, o una red de una manera visualmente agradable.

- *JWNL*: (Java WordNet Library) es una API que permite acceder al diccionario relacional WordNet. También provee funcionalidades más allá del acceso a datos, tales como el descubrimiento de relaciones y procesamiento morfológico.

❖ **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento del prototipo. Esta capa se compone de los siguientes paquetes:

- *Repositorio de Servicios*: Es un repositorio de procesos de negocio, que soporta la recuperación de archivos BPEL (y otros documentos XML) y provee un poderoso mecanismo de consulta que permite encontrar un proceso con búsquedas en sus propiedades o metadatos relacionados. Este repositorio provee una API (paquete *BPEL Repository*) con el fin de hacer flexible su uso.

- *WordNet*: es una enorme base de datos léxica del idioma inglés. Agrupa las palabras en conjuntos de sinónimos llamados 'synsets', proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos. El objetivo es de este proyecto es: por un lado producir una combinación de diccionario y tesauro cuyo uso es más intuitivo, y ayudar al análisis automático de textos y a las aplicaciones de inteligencia artificial. Este paquete es utilizado por el

- *Comparador Lingüístico* (a través del paquete JWNL) para verificar la relación semántica entre dos palabras.

A.2.2.2 Diagrama de Paquetes de la aplicación SeMatch-Context

La Figura A7 presenta la vista lógica de la aplicación SeMatch-Context. A continuación se describen las capas y su interacción, así como los paquetes más relevantes que las componen.

- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades de la aplicación. Ésta capa está compuesta por:

  - *Contexto de Entrega*: éste paquete permite obtener el contexto de entrega de los clientes móviles que acceden a la aplicación, tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc.

  - *Explorador de Capacidades General*: éste paquete permite explorar las capacidades de un dispositivo móvil, obteniendo la información de la biblioteca WURFL o de los repositorios UAProf.

  - *Emparejamiento de Servicios*: éste paquete contiene la lógica que permite recuperar los servicios más relevantes para la consulta de un usuario, mediante una fase emparejamiento a nivel atómico de las actividades BPEL contenidas en el *Repositorio de Servicios*, considerando los requerimientos de contexto en el cual el servicio puede ser invocado.

  - *Analizador BPEL-Grafos*: éste paquete permite registrar funciones que transforman meta-modelos de servicios en su equivalente en grafos. Para el prototipo ServiceMatch se ha implementado las funciones de BPEL. Sin embargo, es posible registrar otro tipo de funciones y extender su aplicabilidad a otros modelos como WSDL o WS-CDL.

- **Capa de Mediación:** contiene las interfaces de programación de aplicaciones (APIs) utilizadas por la aplicación.

  - *DELI*: librería que permite a un Servlet Java conocer el contexto de entrega de un dispositivo a través de CCPP o UAProf. El contexto de entrega ayuda a los servidores a identificar en qué clase de entorno se está realizando la petición y con base en esto puede dar una respuesta más adecuada según el dispositivo.

- *WALL*: *Wireless Abstraction Library* (Librería de Abstracción Móvil) es una librería de tags JSP que permite diseñar páginas Web que, dependiendo de las capacidades del dispositivo, entrega el contenido en formato WML, C-HTML, y XHTML Mobile Profile.

- *Xerces*: es un Parser XML compuesto por una familia de paquetes software que permiten analizar y manipular archivos XML. La biblioteca implementa una serie de APIs estándar que incluye a DOM, SAX y SAX2.

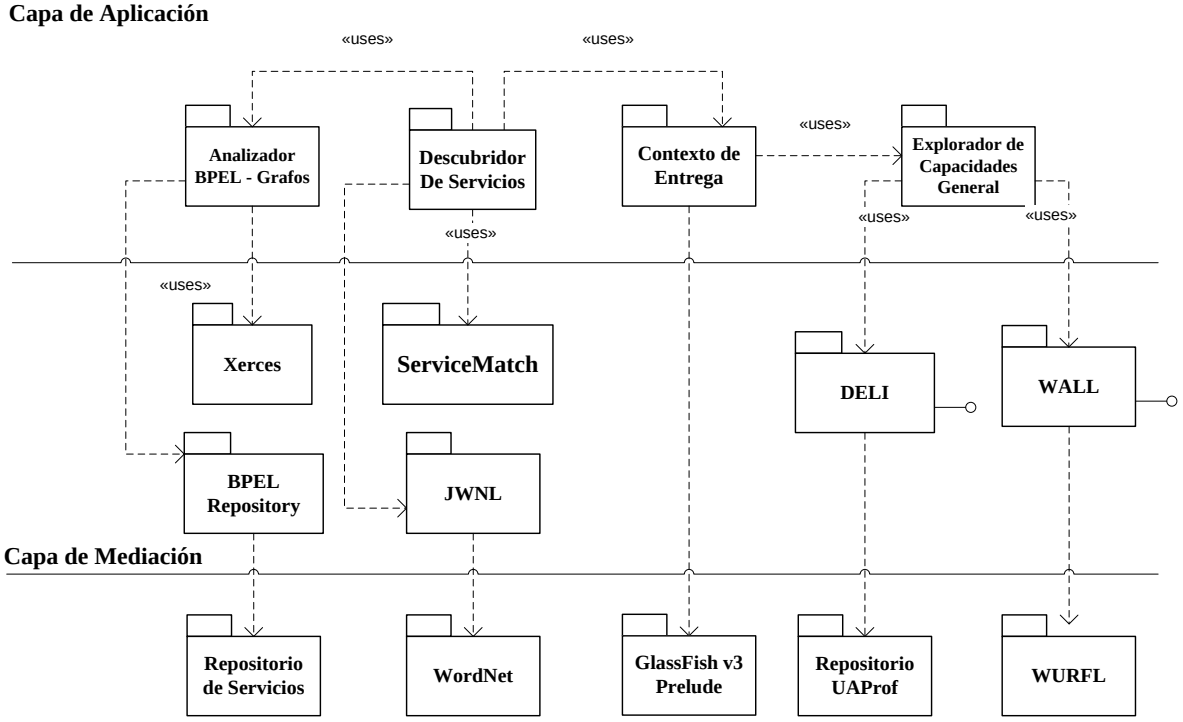


Figura A 7. Vista lógica de la Aplicación SeMatch-Context.

- *JWNL*: (Java WordNet Library) es una API que permite acceder al diccionario relacional WordNet. También provee funcionalidades más allá del acceso a datos, tales como el descubrimiento de relaciones y procesamiento morfológico.

- *BPEL Repository*: es una API de Java que permite manipular documentos BPEL (y otros archivos XML relacionados), encontrados en el *Repositorio de Servicios,* como objetos. Ocultando la serialización y deserialización al usuario. Las consultas son realizadas utilizando un lenguaje de consulta orientado a objetos, OCL.

- *ServiceMatch*: representa el *.JAR* (*Java ARchive*) del prototipo ServiceMatch definido. Los *.jar* son un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Dada la lógica implementada en el *ServiceMatch*, permite determinar la similitud entre un nodo *Query* y una colección de nodos *Target* (1:N), y entregar una lista ordenada de las actividades más similares a un nodo de consulta.

- **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento de la aplicación. Esta capa se compone de los siguientes paquetes:

  - *GlassFish v3Prelude*: es un servidor de aplicaciones open source desarrollado por Sun Microsystems, que implementa las tecnologías definidas en la plataforma Java EE 6 y permite ejecutar aplicaciones que siguen esta especificación.

  - *WURFL*: *Wireless Universal Resource FiLe* (Archivo de Recursos Universal Inalámbrico). Es parte del esfuerzo de una comunidad FOSS (Free and Open Source Software, Código Fuente Libre y Abierto) enfocada en el problema de presentar contenido en la amplia variedad de dispositivos móviles. El WURFL es un fichero de configuración XML que contiene información acerca de características y capacidades para una variedad de dispositivos móviles. Dicha información es contribución de desarrolladores de todo el mundo y el WURFL es actualizado de forma frecuente reflejando los nuevos dispositivos móviles que entran en el mercado

  - *Repositorio UAProf*: es una especificación basada en el CCPP que se encarga de describir las preferencias de usuario y las capacidades de los dispositivos, con el fin de obtener una experiencia de usuario más personalizada.

  - *Repositorio de Servicios*: Es un repositorio de procesos de negocio, que soporta la recuperación de archivos BPEL (y otros documentos XML) y provee un poderoso mecanismo de consulta que permite encontrar un proceso con búsquedas en sus propiedades o metadatos relacionados. Este repositorio provee una API (paquete *BPEL Repository*) con el fin de hacer flexible su uso.

  - *WordNet*: es una enorme base de datos léxica del idioma inglés. Agrupa las palabras en conjuntos de sinónimos llamados '*synsets*', proporcionando definiciones cortas y generales, y almacenando las relaciones semánticas entre estos conjuntos de sinónimos. El objetivo es de este proyecto es: por un lado producir una combinación de diccionario y tesauro cuyo uso es más intuitivo, y ayudar al análisis automático de textos y a las aplicaciones de inteligencia artificial. Este paquete es utilizado por el *Comparador Lingüístico* (a través del paquete JWNL) para verificar la relación semántica entre dos palabras.

A.2.2.3 Diagrama de Paquetes del Repositorio de Usuarios

En la figura Figura 18 se presenta la vista lógica de toda la plataforma U-ServiceMatch, especialmente la parte de personalización, para mostrar su integración con los demás componentes.

- **Capa de Aplicación:** contiene los paquetes que implementan las funcionalidades de la aplicación. Ésta capa está compuesta por:

  - *Service Discovery:* éste paquete contiene la clase ServiceDiscovery, encargada de implementar la lógica que permite recuperar los servicios más relevantes para la consulta de un usuario. Ésta clase, en la última fase de integración, hace uso del módulo de Perfil de Usuario para conseguir un número menor de servicios sugeridos, para ser organizados mediante una fase emparejamiento a nivel atómico. Si estos servicios no tuvieran las características necesitadas, se realiza una búsqueda sobre las actividades BPEL contenidas en el *Repositorio de Servicios.* luego utilizando el módulo de Contexto de Entrega se ejecuta el proceso de descarte y así se obtienen las actividades consumibles por el dispositivo solicitante.

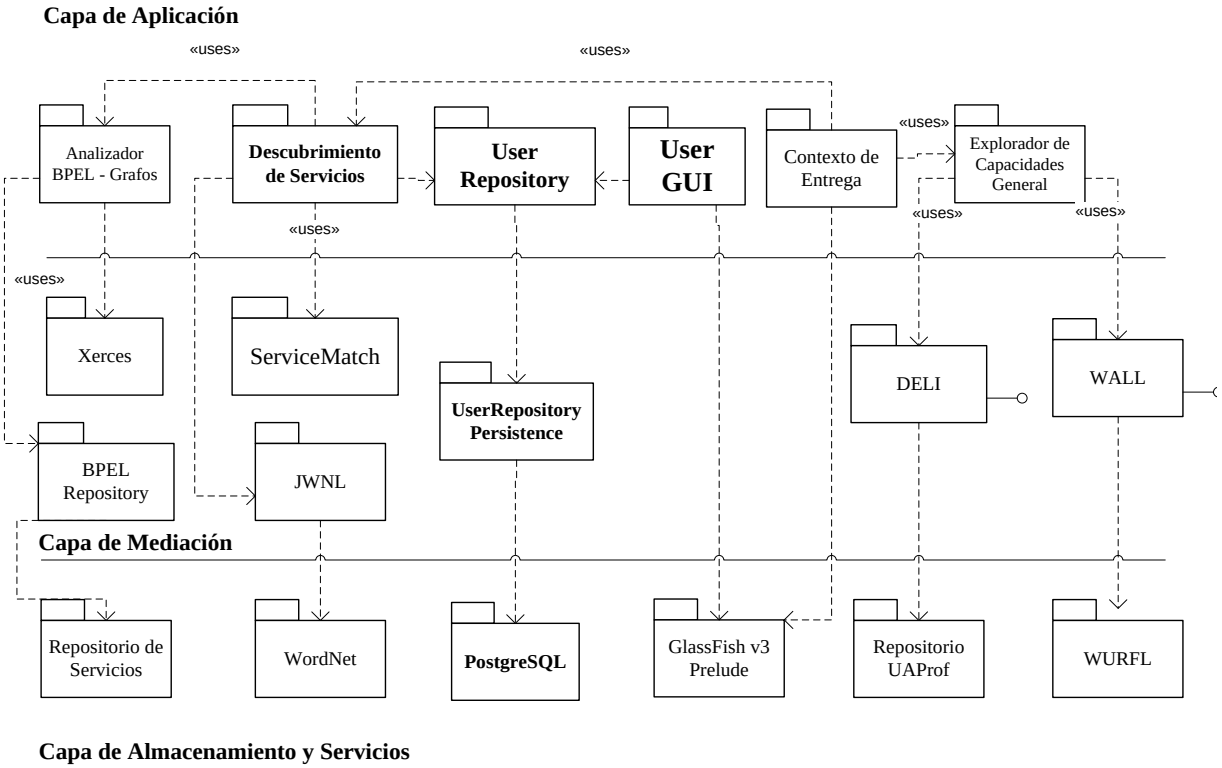


Figura A 8. Diagrama de Paquetes Repositorio de Usuarios

  - *User Repository:* representa el *.JAR* que contiene las clases relacionadas con la lógica del proceso de sugerencia de servicios, éste componente brinda todas las funciones que el módulo de descubrimiento necesita para soportar la personalización

de su proceso. También soporta las operaciones que UserGUI realice sobre la información de usuario.

- *UserGUI: éste* es el componente que ofrece una interfaz al cliente, ésta interfaz le permite registrar sus datos personales (perfil de usuario)

❖ **Capa de Mediación:** contiene las interfaces de programación de aplicaciones (APIs) utilizadas por la aplicación.

- *UserRepositoryPersistence*: es el componente encargado de brindar el soporte a la persistencia de los datos de usuario.

❖ **Capa de Almacenamiento y Servicios:** incluye el software básico que permite el funcionamiento de la aplicación. Esta capa se compone de los siguientes paquetes:

- *PostgreSQL:* es un sistema de gestión de bases de datos objeto-relacional (ORDBMS), utilizado para generar las tablas, que contienen la información relacionada con el usuario.

A.2.3 Descripción Detallada de las Clases del Sistema

A.2.3.1 Descripción Detallada de las Clases de ServiceMatch

En ésta sección se describirán detalladamente las clases que consideramos más relevantes para la funcionalidad de la aplicación ServiceMatch.

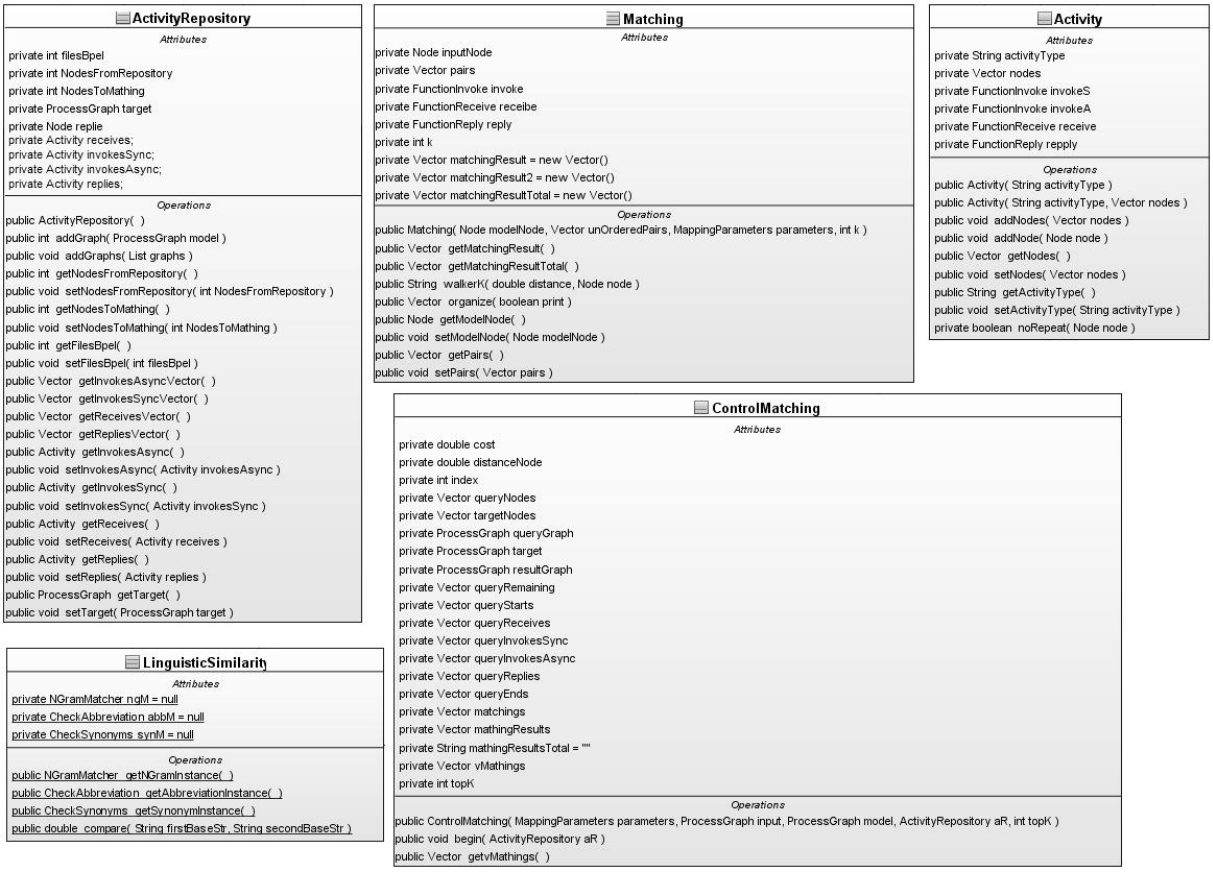


Figura A 9. Diagrama detallado de las Clases más importantes de ServiceMatch

ActivityRepository: ésta clase contiene la lógica para obtener las funciones/nodos de los Grafos de procesos, para organizarlos y almacenarlos por tipo de actividad (*invoke, reply, receive*).

Activity: ésta clase representa la actividad/nodo/función de un Grafo. La cual puede ser: FunctionInvoke, FunctionReceive, FunctionReply. Cada función posee cuatro atributos: Tipos de Actividad, Nombre de la operación PartnerLink y PortType.

Matching: está clase toma como entradas dos nodos, que representan actividades básicas de BPEL (receive, invoke, reply), y calcula la distancia semántica entre ellos. Para esto utiliza las clases: InvokeDistanceNode, ReceiveDistanceNode y ReplyDistanceNode; quienes a su vez usan la clase LinguisticSimilarity.

A.2.3.2 Descripción Detallada de las Clases de SeMatch-Context

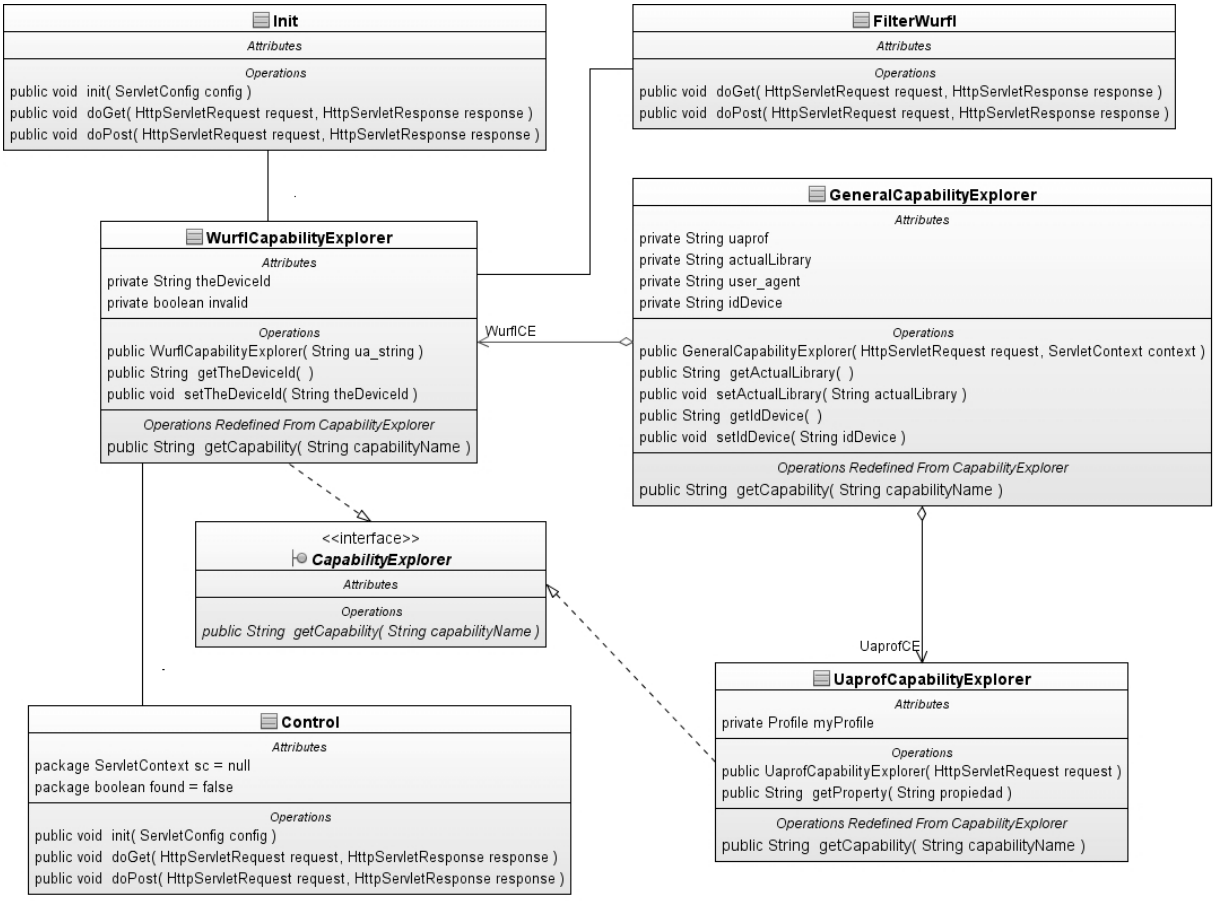


Figura A 10. Diagrama detallado de las Clases más importantes de SeMatch-Context

Nombre	Descripción
GeneralCapabilityExplorer	Ésta clase permite explorar las capacidades de un dispositivo móvil, obteniendo la información de la biblioteca WURFL o de los repositorios UAProf.
Atributos	
▪ **uaprof**: Cadena que contiene la URL del repositorio uaprof propio del dispositivo. ▪ **UaprofCE**: Explorador de capacidades en los repositorios uaprof. ▪ **WurflCE**: Explorador de capacidades para la biblioteca WURFL. ▪ **actualLibrary**: Cadena que identifica la fuente de información actual, puede tomar los valores “uaprof” o “wurfl”. ▪ **user_agent**: Cadena que contiene la cabecara HTTP user_agent.	
Métodos	
▪ **getCapability**: Retorna cierta capacidad del dispositivo de la fuente de información actualmente activa. ▪ **getActualLibrary**: Retorna la fuente de información para extraer las capacidades del dispositivo. ▪ **setActualLibrary**: Cambia la fuente información para extraer las capacidades del dispositivo.	

Figura A 11. Descripción detallada de la Clase GeneralCapabilityExplorer

Nombre	Descripción
WurflCapabilityExplorer	Ésta clase se encarga de explorar las capacidades de un móvil obteniendo la información en la biblioteca WURFL.
Atributos	
▪ **theDeviceId:** Número que identifica el dispositivo dentro de la biblioteca WURFL. ▪ **invalid:** Bandera que indica si el dispositivo se encuentra en la biblioteca WURFL**.**	
Métodos	
▪ **getCapability:** Retorna cierta capacidad del dispositivo. ▪ **getTheDeviceId:** Retorna el número que identifica el dispositivo dentro de la biblioteca WURFL. ▪ **setTheDeviceId:** Cambia el número que identifica el dispositivo dentro de la biblioteca WURFL**.**	

Figura A 12. Descripción detallada de la Clase WurflCapabilityExplorer

Nombre	Descripción
UAProfCapabilityExplorer	Ésta clase se encarga de explorar las capacidades de un móvil obteniendo la información en los repositorios UAProf.
Atributos	
▪ **myProfile:** Objeto que contiene la información de las capacidades del dispositivo.	
Métodos	
▪ **getCapability:** Retorna cierta capacidad del dispositivo. ▪ **getTheDeviceId:** Retorna el número que identifica el dispositivo dentro de la biblioteca WURFL. ▪ **setTheDeviceId:** Cambia el número que identifica el dispositivo dentro de la biblioteca WURFL**.**	

Figura A 13. Descripción detallada de la Clase UAProfCapabilityExplorer.

Nombre	Descripción
CapabilityExplorer	Interfaz que define la estructura de los exploradores de capacidades para dispositivos móviles.
Atributos	
▪ Ninguno	
Métodos	
▪ **getProperty:** Retorna una propiedad del dispositivo. ▪ **getCapability:** Retorna cierta capacidad del dispositivo.	

Figura A 14. Descripción detallada de la Clase CapabilityExplorer

Nombre	Descripción
Control	Clase encarga de controlar la obtención de información en las bibliotecas WURFL.
Atributos	
▪ **sc:** Objeto que tiene la información del contexto donde se encuentra ubicado el servidor propio de la librería WALL. ▪ **found:** Bandera que indica si un objeto a sido encontrado en la librería.	
Métodos	
▪ **init:** Inicializa la clase. ▪ **doGet:** Atiende las peticiones a la librería WURFL por el método Get. ▪ **doPost:** Atiende las peticiones a la librería WURFL por el método Post.	

Figura A 15. Descripción detallada de la Clase Control

Nombre	Descripción
Init	Se encarga de cargar en RAM el archivo “wurfl.xml” que contiene la información de la biblioteca WURFL.
Atributos	
▪ **init:** Inicializa la clase y la librería WALL. ▪ **doGet:** Atiende las peticiones a la librería WALL por el método get. ▪ **doPost:** Atiende las peticiones a la librería WALL por el método post.	
Métodos	
▪ **init:** Inicializa la clase. ▪ **doGet:** Atiende las peticiones a la librería WURFL por el método Get. ▪ **doPost:** Atiende las peticiones a la librería WURFL por el método Post.	

Figura A 16. Descripción detallada de la Clase Init.

Nombre	Descripción
FilterWurfl	Buscador de las capacidades de un dispositivo en la biblioteca WURFL mediante la cabecera HTTP, “User-Agent”.
Atributos	
▪ Ninguno	
Métodos	
▪ **doGet:** Atiende las peticiones de búsqueda de capacidades por el método Get. ▪ **doPost:** Atiende las peticiones de búsqueda de capacidades por el método Post.	

Figura A 17. Descripción detallada de la Clase FliterWurfl.

A.3 Fase de Implementación y puesta a punto

En ésta fase se describe la implementación de los prototipos operacionales que integran las la Plataforma para el descubrimiento de Servicios en Ambientes Ubicuos U-ServiceMatch. Se tomará en consideración la guía metodológica RUP (Rational Unified Process) para la implementación de los prototipos definidos (iteraciones proyectadas): ServiceMatch, SeMatch-Context y Repositorio de Usuarios.

Inicialmente, se detalla las funcionalidades, arquitectura e interfaz del Matching Básico de Servicios (ServiceMatch), el cual está orientado al emparejamiento, a nivel atómico, de actividades BPEL sin considerar las características de ubicuidad. Luego se presenta el repositorio de Capacidades de Dispositivos. Para finalizar esta sección, el Repositorio de Usuarios es expuesto. Sus funcionalidades, así como las arquitectura e interfaces de usuario son detalladas.

A.3.1 Prototipo 1 - Matching Básico de Servicios: ServiceMatch

Se desarrolló un prototipo que implementa una técnica de emparejamiento, a nivel atómico, de actividades BPEL que trabaja sobre un repositorio de procesos de negocio (Vanhatalo, y otros, 2006.). Su implementación fue realizada usando Java como lenguaje de programación (JDK 1.6.0) y Netbeans 6.7 como entorno de desarrollo (IDE). El matching básico de servicios, denominado ServiceMatch, es una aplicación de escritorio que toma como entrada una actividad BPEL (*receive, invoke, reply*), calcula la similitud entre las actividades del mismo tipo contenidas en el repositorio de procesos, y finalmente entrega una lista ordenada de las actividades más similares recuperadas.

En el resto de esta sección, en primer lugar se describen las funcionalidades del sistema; luego, su arquitectura y, finalmente, se presenta las interfaces de usuario proporcionadas por el prototipo.

A.3.1.1 Funcionalidades del Sistema

Dado un conjunto de servicios publicados con funcionalidades equivalentes (correspondientes a un dominio dado, por ejemplo, servicios de reserva de viajes), el objetivo de este prototipo es clasificar los servicios recuperados, considerando su idoneidad con respecto a los requerimientos del usuario. Se parte del supuesto que el usuario expresa sus necesidades mediante un servicio representado como un modelo de comportamiento y la plataforma le ayudará a identificar los servicios más similares. En éste primer prototipo el ranking de servicios se basa en un emparejamiento a nivel atómico de las actividades contenidas en el repositorio de procesos de negocio. El sistema es presentado en la Figura A16.

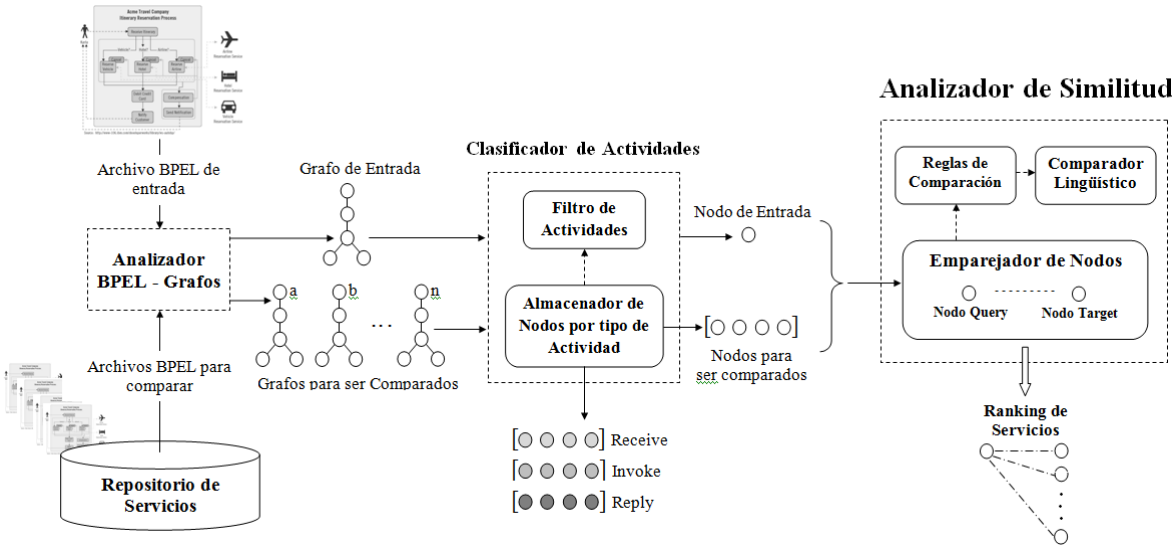


Figura A 18. Plataforma para el Emparejamiento de Actividades BPEL: ServiceMatch

- *Analizador BPEL - Grafos:* éste módulo transforma las descripciones de comportamiento (BPEL) en su equivalente en grafos.

- *Filtro de Actividades*: éste módulo se encarga de verificar, filtrar y confirmar que las actividades (obtenidas de los grafos que representan a los procesos publicados en el Repositorio) a ser comparadas no estén repetidas.

- *Almacenador de Nodos por tipo de Actividad*: éste módulo obtiene, almacena y clasifica los nodos por tipo de actividad (*receive, invoke y reply*). Durante este proceso utiliza el *Filtro de Actividades* para evitar que nodos iguales sean almacenados y así, agilizar el proceso de emparejamiento.

- *Reglas de Comparación*: Este módulo contiene las funciones que permite calcular la distancia semántica entre dos actividades, para esto usa el *Comparador Lingüístico*.

- *Comparador Lingüístico*: éste componente calcula la similitud lingüística entre dos cadenas usando los siguientes algoritmos. *NGram, Check synonym* y *Check abbreviation*.

- *Emparejador de Nodos*: este módulo contiene la lógica que permite comparar y establecer la distancia semántica entre dos nodos (*Query* y *Target*), para éste proceso usa las *Reglas de Comparación* y el *Comparador Lingüístico*. Una vez concluidas todas las comparaciones entre el nodo de entrada y los nodos de los grafos abstraídos del repositorio de procesos, entrega un ranking organizado de los servicios con la menor distancia o mayor similitud al nodo de consulta.

- *Repositorio de Servicios*: Es un repositorio de procesos de negocio, que soporta la recuperación de archivos BPEL (y otros documentos XML) y provee un poderoso mecanismo de consulta que permite encontrar un proceso con búsquedas en sus propiedades o metadatos relacionados. Este repositorio provee una API (paquete *BPEL Repository*) con el fin de hacer flexible su uso.

A.3.1.2 Interfaz de Usuario

En esta sección, se presenta la interfaz gráfica (GUI) de la plataforma ServiceMatch, Figura A17. La figura muestra el emparejamiento del nodo *Query* con la colección de nodos *Target* del mismo tipo. El resultado arroja una lista (*Top K*) de los nodos más similares al de consulta, organizada según la distancia semántica encontrada.

El proceso de emparejamiento comienza cuando el usuario carga el archivo BPEL, entonces se ejecuta el *Analizador BPEL-Grafos* que transforma en grafos tanto el documento de entrada, como los procesos publicados en el *Repositorio de Servicios*. Luego, la plataforma ServiceMatch permite asignar pesos a los atributos: *Operation*, *Partnerlik* y *PortType* de los nodos, según el grado de relevancia considerado por el usuario, y el *Top K,* que es el número de actividades que desea emparejar. Finalmente el resultado del emparejamiento es presentado al usuario, en el se puede observar: el tipo de nodos (*receive*, *invoke* y *reply*) emparejados, el *nodo Query*, el conjunto de *nodos Target* y la lista del top K de nodos organizados según la distancia semántica encontrada.

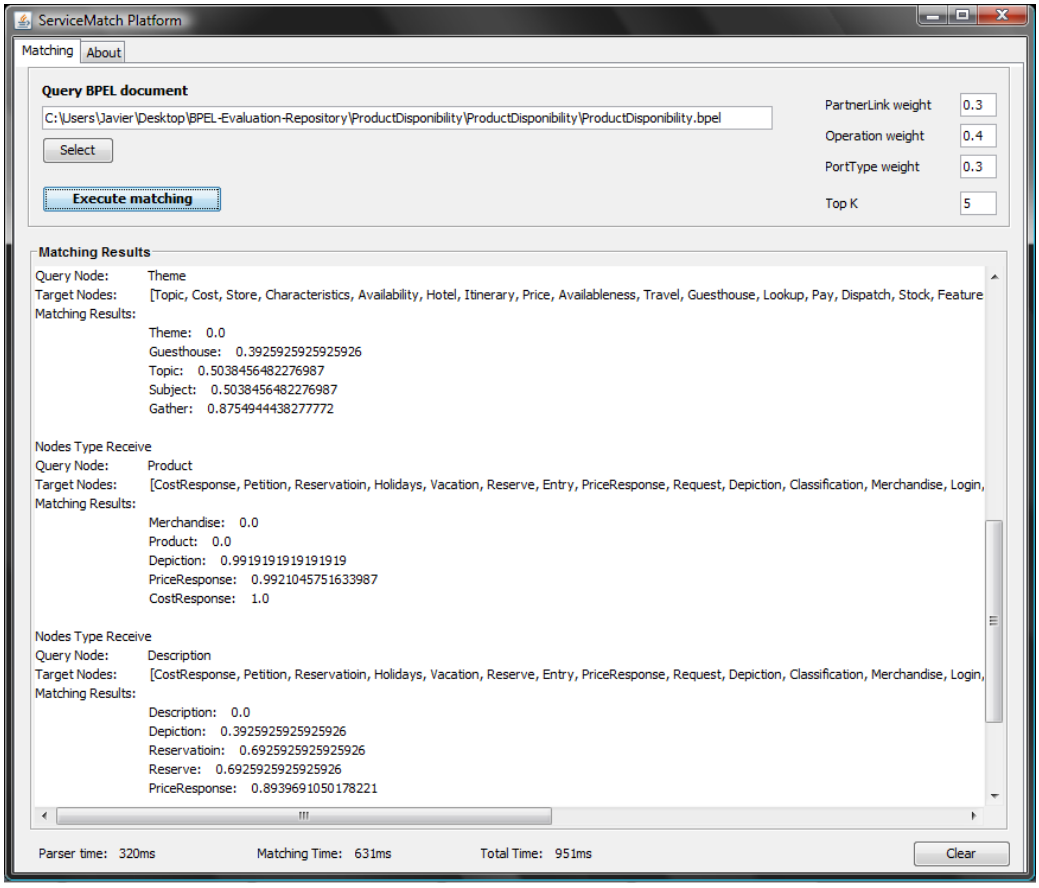


Figura A 19. Interfaz de emparejamiento de ServiceMatch

A.3.2 Prototipo 2 – Aplicación SeMatch-Context

Éste prototipo corresponde a la integración del Repositorio de Dispositivos con ServiceMatch. Aplicación en la cual se implementó un mecanismo de consulta, que permite obtener los requerimientos de contexto de un determinado servicio.

A.3.2.1 Repositorio de Dispositivos: DeviceCapabilities

Considerando las características y funcionalidades ofrecidas por WURFL y UAProf, los cuales son ampliamente descritos en el ANEXO B, se definió un Repositorio que permite obtener las capacidades de un gran número de dispositivos móviles. Su implementación fue realizada sobre Glassfish V2.1 con J2EE (versión 1.4) y Netbeans 6.7 como entorno de desarrollo (IDE). Éste prototipo es una aplicación Web denominada *DeviceCapabilities* que permite obtener el contexto de entrega, el cual incluye las capacidades de cada dispositivo cliente. El contexto de entrega es construido con base en tres fuentes de información UAProf, WURFL y las cabeceras HTTP, lo que garantiza una descripción amplia y fiel de las capacidades del dispositivo.

En el resto de esta sección, en primer lugar se describen las funcionalidades del sistema; luego su arquitectura y finalmente, la experimentación de la aplicación es presentada.

Funcionalidades del Sistema

El objetivo del prototipo DeviceCapabilities es determinar el contexto de entrega de un dispositivo cliente. Como se mencionó en el capítulo 4, la gestión del contexto para éste trabajo de grado se considera dos variables: *i*) *requerimientos de contexto del servicio*, manejado por el *Repositorio de Servicios* y *ii) restricciones del contexto del solicitante* que son las características software y hardware de los dispositivos que solicitan los servicios (capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc.). La aplicación DeviceCapabilities se encarga de definir ésta segunda variable por medio del contexto de entrega.

El prototipo inicialmente usa la información que envía el navegador por medio de las cabeceras HTTP, luego los datos ofrecidos por la librería WURFL y por último usa UAProf. Se ha considerado que éste es el orden de fidelidad de la información, ya que el navegador debe conocer fielmente sus capacidades y la librería WURFL ha sido construida por desarrolladores que mediante pruebas obtienen éstos datos; mientras que en el caso de UAProf suele obtenerse información que no detalla el producto, sino una categoría de dispositivos del fabricante. Tal como se observa en el diagrama de secuencia presentado en la Figura A18.

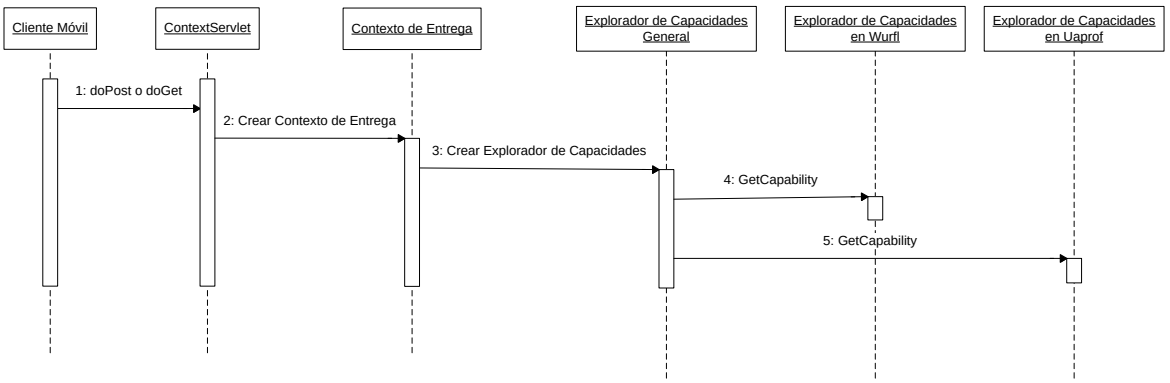


Figura A 20. Diagrama de Secuencia para obtener el contexto de entrega

Diferentes dispositivos móviles acceden a la aplicación a través del protocolo HTTP, en el cual se añade una descripción de su contexto de entrega mediante el estándar CCPP; esto le permite a la aplicación extraer toda la información relacionada con el móvil de la librería WURFL y el repositorio UAProf respectivamente. Lo anterior se puede observan en la Figura A19.

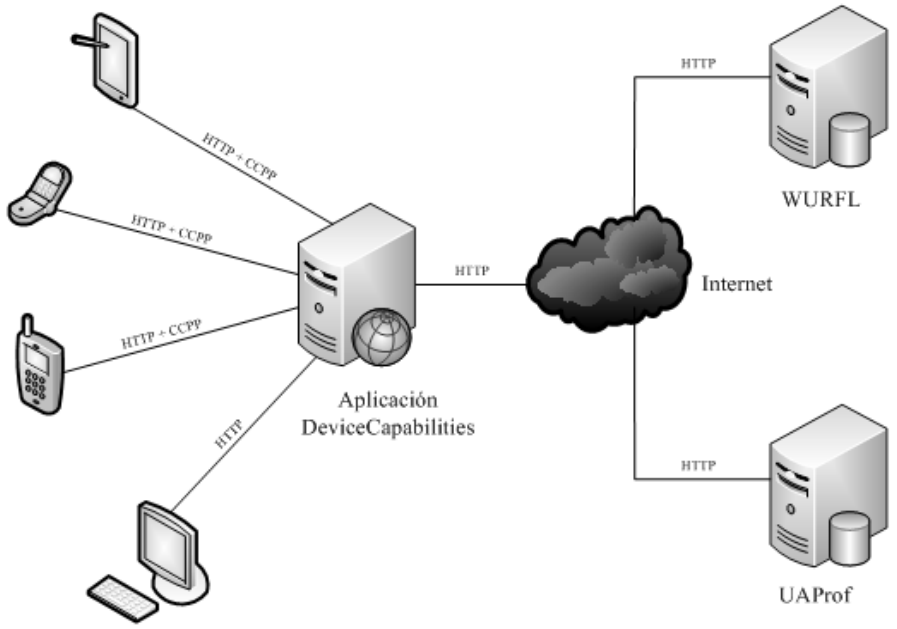


Figura A 21. Aplicación para obtener el contexto de entrega: DeviceCapabilities

A.3.2.2 Mecanismo de Consulta de la aplicación ServiceMatch

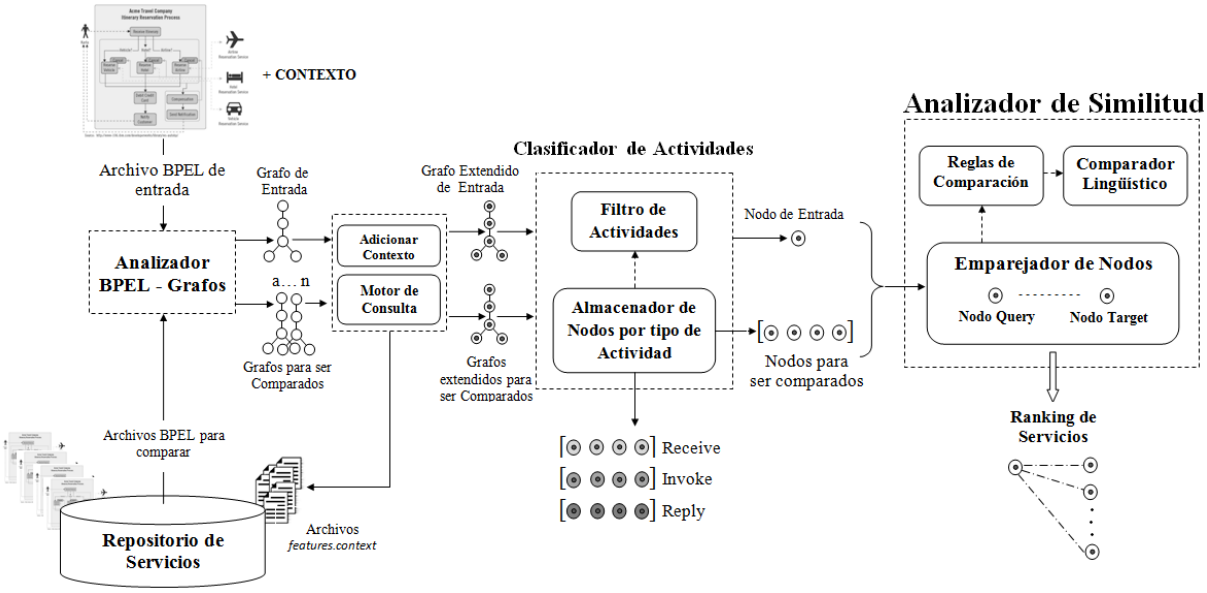


Figura A 22. Mecanismo de Consulta de Requerimientos de Contexto para la Plataforma ServiceMatch.

El mecanismo de consulta implementado en ServiceMatch es soportado por dos módulos descritos a continuación:

KentOCLQueryEngineImpl: implementa el motor de consulta principal de la versión pública del *Repositorio de Servicios* desarrollado por la Universidad de Kent. Es un motor de consulta OCL de código abierto que implementa el estándar OCL 2.0. Para éste prototipo, el motor OCL de Kent obtiene las URI de los archivos BPEL almacenados en el repositorio de procesos de negocio.

Adicionar Contexto: éste módulo aprovecha las características de flexibilidad que la representación formal de grafos ofrece. Una vez el *Motor de Consulta* obtiene el contexto del servicios del archivo *features.context* asociado al documento BPEL, éste adiciona el atributo *Access Type* al grafo; haciendo que los nodos que lo componen posean un parámetro mas. (*Operation, PortType, PartnerLink* y *AccessType*).

La fase de emparejamiento no se ve afectada, dado que sigue comparando los mismos atributos.

En la Figura A21 se presenta los resultados del emparejamiento de manera textual en el, mediante la interfaz de la aplicación ServiceMatch v2.0 donde se muestra el tipo de actividad de los nodos emparejados, el nodo de consulta, la colección de los nodos del repositorio de servicios y el resultado presentado como una lista ordenada de los nodos más relevantes recuperados para un top K, junto con el valor de similitud y el atributo de contexto. Además, la interfaz permite al usuario fijar los pesos Wop, Wpl y Wpt, e ingresar el requerimiento del contexto, Tipo de Acceso (por ejemplo, GSM, CDMA, etc.). Con esto experimentamos el mecanismo de consulta implementado.

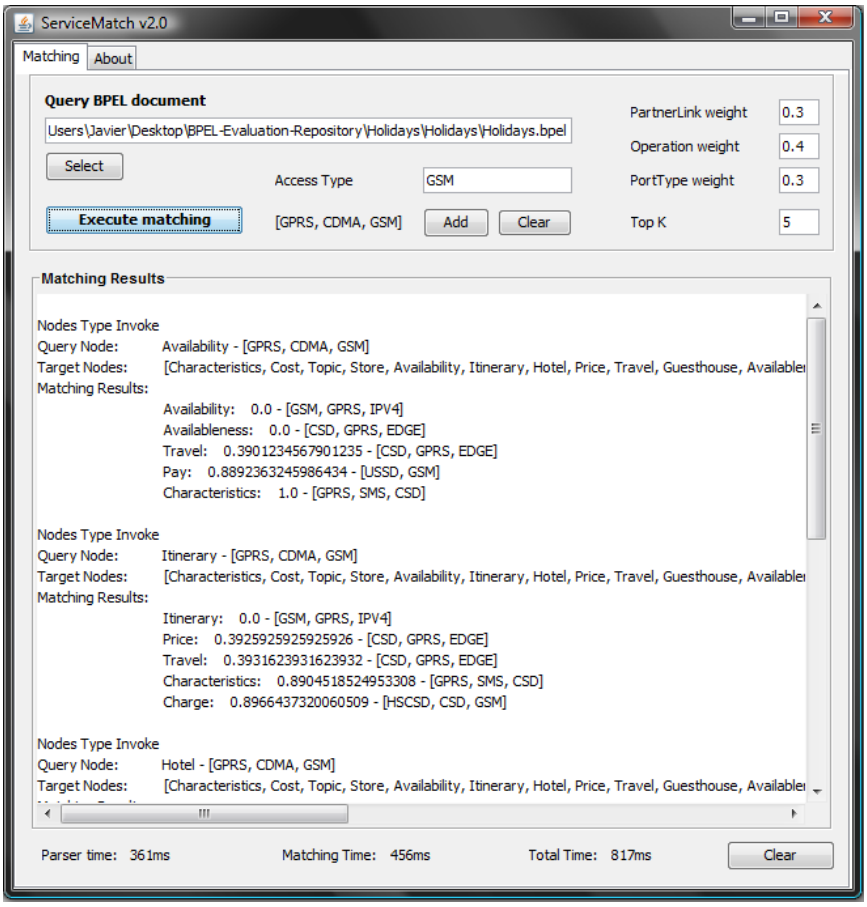


Figura A 23. Interfaz de emparejamiento de ServiceMatch con el mecanismo de Consulta del Contexto.

ANEXO B

REPOSITORIOS

B.1 Repositorio para Procesos de Negocios BPEL y metadatos arbitrarios asociados

El Repositorio presentado en (Vanhatalo, y otros, 2006) está diseñado para trabajar sobre archivos BPEL y otros archivos relacionados con ellos. El proyecto no pretende ser un repositorio XML genérico multipropósito, pero está construido para soportar entornos centrados en BPEL.

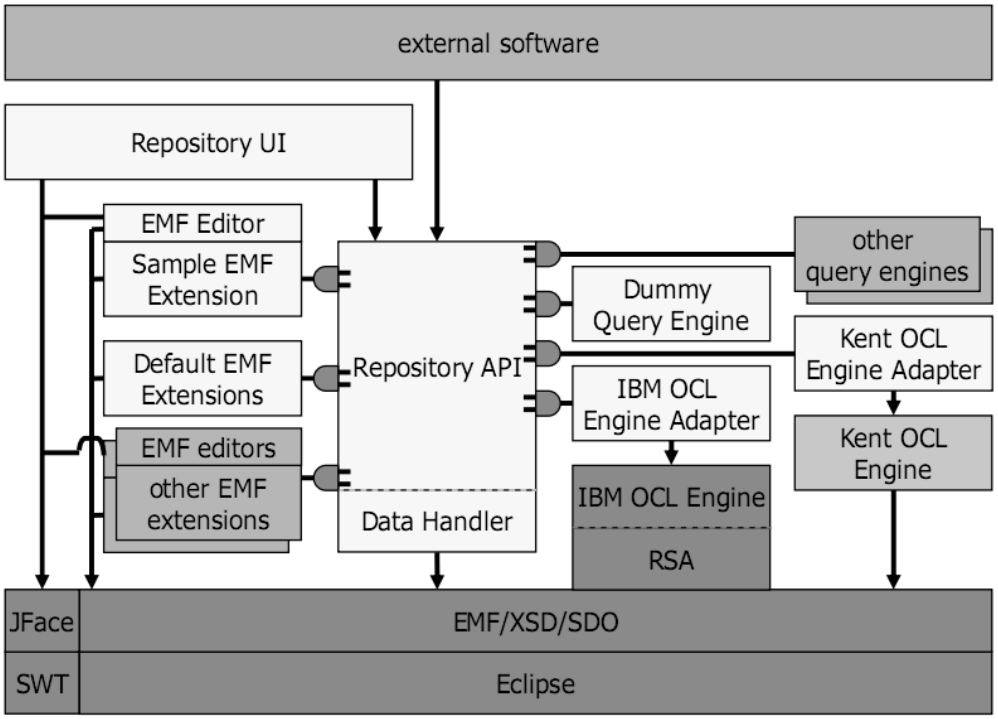


Figura A 24. Arquitectura del Repositorio BPEL.

La arquitectura del repositorio BPEL es presentada en la Figura A22. Los componentes azules son proporcionados por *Eclipse Foundation* (Eclipse, 2009). Los componentes en amarillo constituyen parte de la solución del repositorio BPEL. Los componentes en verde muestra las diferentes formas de software externo que puede ser implementado para extender o usar el repositorio. Todos los componentes son *plug-ins* de Eclipse. El componente fundamental de esta solución es el *Rository API*, que proporciona una interfaz para software externo. La interfaz de usuario del repositorio (UI) es una aplicación de ejemplo que utiliza el *Repository API*. Sin embargo, también es una interfaz grafica de usuario muy útil para gestionar el contenido del repositorio. La interfaz de usuario está integrada en el Eclipse *workbench* y construida con SWT (Standard Widget Toolkit) y librerías JFace. El Data Handler es un subcomponente que se encarga del acceso a datos y del sistema de archivos. El Data Handler usa el EMF (Eclipse Modeling Framework) (Budinsky, y otros, 2004) para serializar objetos EMF en archivos XML y de-serializarlos

nuevamente a objetos EMF. De esta forma todos los componentes del repositorio manipulan datos como objetos EMF en lugar de XML.

B.1.1 Flexibilidad para manipular los datos como objetos EMF

En el repositorio los datos son representados como objetos EMF. El modelo EMF puede ser generado automáticamente desde un esquema XML, un diagrama de clases UML o clases Java (Budinsky, y otros, 2004) (Eclipse, 2009). En el contexto de gestión de procesos de negocio los datos son frecuentemente almacenados como XML, conforme a los estándares. El repositorio puede ser extendido para soportar un nuevo tipo de dato pero incluyendo su respectivo modelo EMF. El *plug-in Default EMF Extensions* contiene modelos EMF para BPEL, WSDL y esquemas XML estándar. Permitiendo que el repositorio soporte este tipo de archivos por defecto. Este componente puede ser remplazado para soportar tipos de archivos completamente diferentes, ya que se usa mecanismos de *plug-in* de Eclipse, sin ninguna modificación en otra parte del repositorio. Similarmente, otras extensiones EMF pueden ser incluidas en el repositorio.

Actualmente, en este tipo de investigaciones, la extensibilidad es una constante; ejemplo de ello es la combinación de la Gestión de Procesos de Negocio con la Semántica. El repositorio puede ser fácilmente extendido para soportar un nuevo tipo de documento conteniendo metadatos o descripciones semánticas relacionadas con dicho proceso de negocio. Lo anterior es uno de los casos que este proyecto de grado aborda, es por ello que este repositorio cumple con todas la exigencias del presente proyecto de investigación.

B.1.2 Motores de Consulta

El repositorio BPEL (Vanhatalo, y otros, 2006) utiliza motores de consulta existentes, algunos de estos son: El motor de consulta que trabaja sobre IBM Rational Software Architect (RSA). Sin embargo éste motor es comercial, es por ello que se acopla al repositorio usando un adaptador (IBM OCL engine adapter). Otro motor OCL fue implementado en la Universidad de Kent (Akehurst, y otros, 2004), el cual es una herramienta *open-soure*, que puede ser incluida en el repositorio usando el Kent OCL Engine Adapter. El Dummy Query Engine es una implementación ejemplo para mostrar como un nuevo motor de consulta puede perfectamente ser integrado al repositorio. Así, motores de consulta basados en otros tipos lenguajes pueden ser usados.

El repositorio mantiene la interacción sobre los objetos consultados, pero cada sub-consulta es ejecutada por un motor, que ha sido previamente incluido (*plugged*) en el repositorio. El repositorio no tiene conocimiento del lenguaje de consulta que está usando. Porque el repositorio simplemente pasa la consulta y otros parámetros desde la interfaz de usuario o un software externo, al motor de consulta seleccionado, junto con el objeto EMF solicitado. Por lo tanto, cualquier motor que pueda realizar consultas sobre los objetos EMF puede ser incluido en el repositorio BPEL de IBM (Vanhatalo, y otros, 2006).

B.1.3 Estructura de Datos

Los datos se organizan jerárquicamente en forma de árbol (Vanhatalo, y otros, 2006). Las organizaciones son mapeadas a directorios en un sistema de archivos, como se puede

observar en la figura 2. Cada organización puede contener un proceso de negocio y de metadatos asociados. Además de estos documentos de datos, un documento descriptor es almacenado en cada organización. Este contiene los tipos de archivos y el rol de cada uno de los documento de datos. Esta información es usada para hacer la conversión entre objetos EMF y archivos XML. Además, el rol describe cómo los documentos de datos están relacionados con los otros documentos en la organización. Por ejemplo, un archivo WSDL almacenado con el repositorio puede contener la interfaz pública de los *partner links* de los procesos de negocios BPEL.

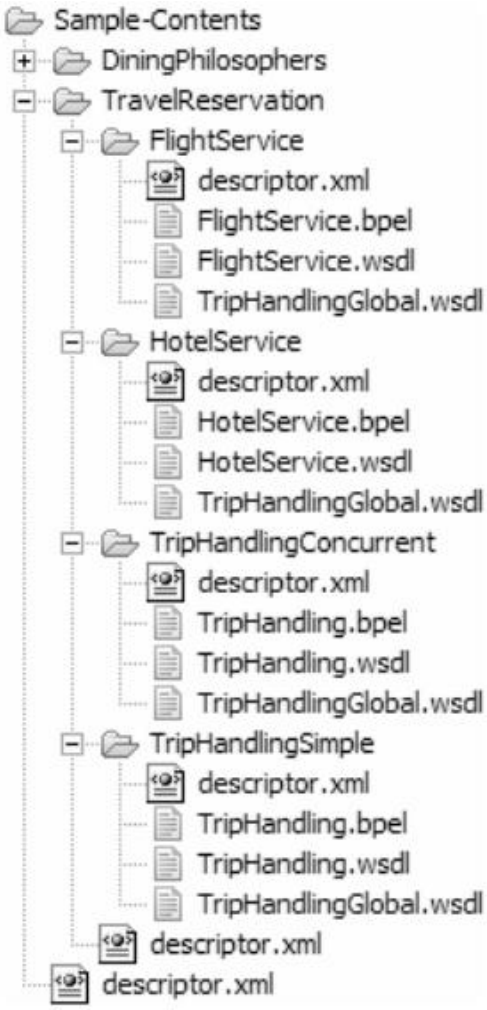


Figura A 25. Un ejemplo de la organización en árbol del Repositorio BPEL

Las consultas se pueden aplicar a los archivos con un rol especificado en una organización, una lista de organizaciones, o una lista de sub-árboles en la jerarquía de la organización. Los archivos relacionados se pueden buscar según sus roles. Los datos se pueden acceder desde el sistema de archivos como archivos XML y por medio del repositorio como objetos EMF. Cualquier directorio en un sistema de archivos puede actuar como la organización raíz de los contenidos del repositorio. Los datos en el repositorio pueden ser trasladados a otro lugar o equipo tan simple como copiar los directorios.

Archivo Descriptor. Cada organización tiene un archivo descriptor XML (descriptor.xml), el cual especifica como los archivos XML están relacionados con el documento BPEL en el folder Organización, en la figura 3 se puede observar como es la organización y relación de los diferentes documentos encontrados en una organización (folder).

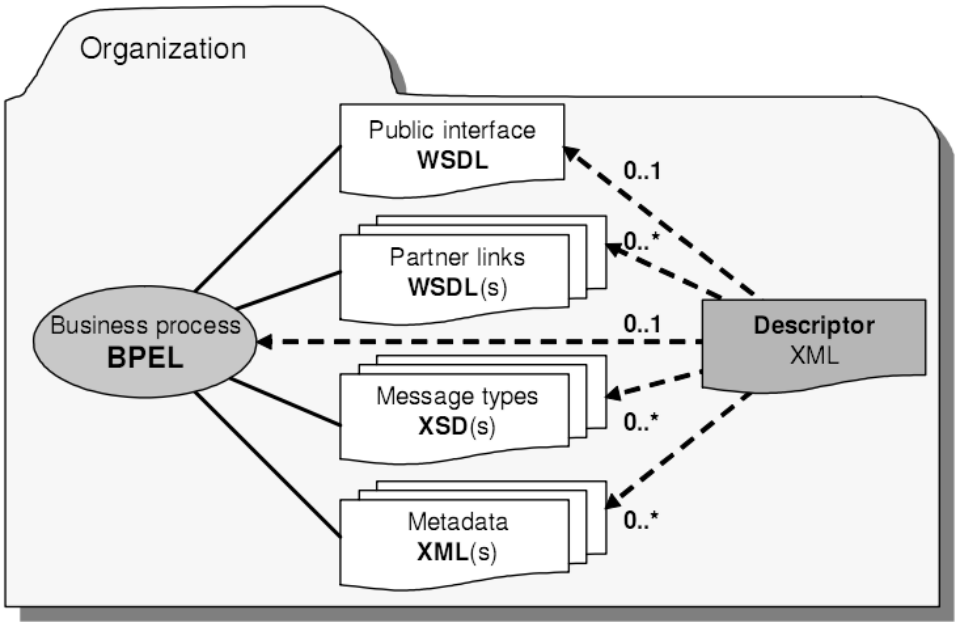


Figura A 26. Archivo descriptor, que considera los vínculos entre los archivos y el documento BPEL de la misma organización especificando sus roles.

El documento XML presentado en la ilustración 1 es un ejemplo de un archivo descriptor. *Sample-contents*. Este incluye un archivo BPEL y una interface pública WSDL para el proceso. Adicionalmente hay otro archivo de interface pública WSDL del proceso asociado. La información del archivo descriptor es también incluida en sí mismo.

Para la organización, un archivo descriptor especifica:

- El nombre del Folder - identificador de la organización en su organización principal.
- URI de Folder - identificador relativo para la raíz del repositorio formando una estructura en árbol (jerarquía).

Para cada archivo en la organización, el archivo descriptor especifica:

- Nombre del Archivo - identificador del archivo en su organización.
- Tipo de Archivo - Es usado para limitar el número de documentos con el mismo tipo de archivo en una organización. Hay cinco diferentes opciones para el tipo de archivo: bpel, wsdl, xsd, metadato, y descriptor.

  En una organización se puede incluir:
  a. 1 documento generado automáticamente con el tipo de archivo descriptor.
  b. 0-1 documentos con el tipo de archivo BPEL.
  c. 0-1 documento con el tipo de archivo WSDL y el tipo de contenido PublicInterface.
  d. 0-n documentos con el tipo de archivo WSDL y el tipo de contenido PartnerLinks.
  e. 0-n documentos con el tipo de archivo XSD.
  f. 0-n documentos con el tipo de archivo METADATO.

```
<?xml version="1.0" encoding="ASCII"?>
<emf:EDescriptor xmlns:emf="http://com/ibm/bpia/repository/emf"
   folderName="FlightService"
   folderURI="Sample-Contents/TravelReservation/FlightService">
  <process name="FlightService.bpel"
     uri="Sample-
     Contents/TravelReservation/FlightService/FlightService.bpel"
     fileType="bpel" contentType="bpel"/>/>
  <descriptor name="descriptor.xml"
     uri="Sample-
     Contents/TravelReservation/FlightService/descriptor.xml"
     fileType="xml" contentType="descriptor"/>
  <publicInterface name="FlightService.wsdl"
     uri="Sample-
     Contents/TravelReservation/FlightService/FlightService.wsdl"
     fileType="wsdl" contentType="PublicInterface"/>
  <partnerLinks name="TripHandlingGlobal.wsdl"
     uri="Sample-
     Contents/TravelReservation/FlightService/TripHandlingGlobal.wsdl"
     fileType="wsdl" contentType="PartnerLinks"/>
</emf:EDescriptor>
```

Figura A 27. Ejemplo del un archivo descriptor usado en el repositorio

- Tipo de Contenido del Archivo - Identificador entre los archivos de la misma organización con diferentes roles y a menudo diferentes modelos EMF. Es utilizado para determinar el modelo EMF usado en el proceso de serialización del archivo. Éste permite el mapeo un modelo EMF a un esquema XML.

- URI - identificador relativo para la raíz del repositorio formando una jerarquía en estructura de árbol.

B.1.4 Mecanismos de Consulta

El repositorio BPEL (Vanhatalo, y otros, 2006) proporciona un mecanismo de consulta, que permite consultar el modelo-objeto de los ficheros en el repositorio. Esto es ejecutado cargando un archivo de consulta en memoria como un objeto EMF. Los objetos EMF son manipulados por un motor de consulta junto con los parámetros de consulta. El motor ejecuta la consulta para el objeto y retorna el resultado al componente *Repository API*. El componente Repository API continúa este proceso interactuando a través de todos los archivos relevantes dentro del ámbito de consulta especificada. El proceso de consulta es ilustrado en la figura 4.

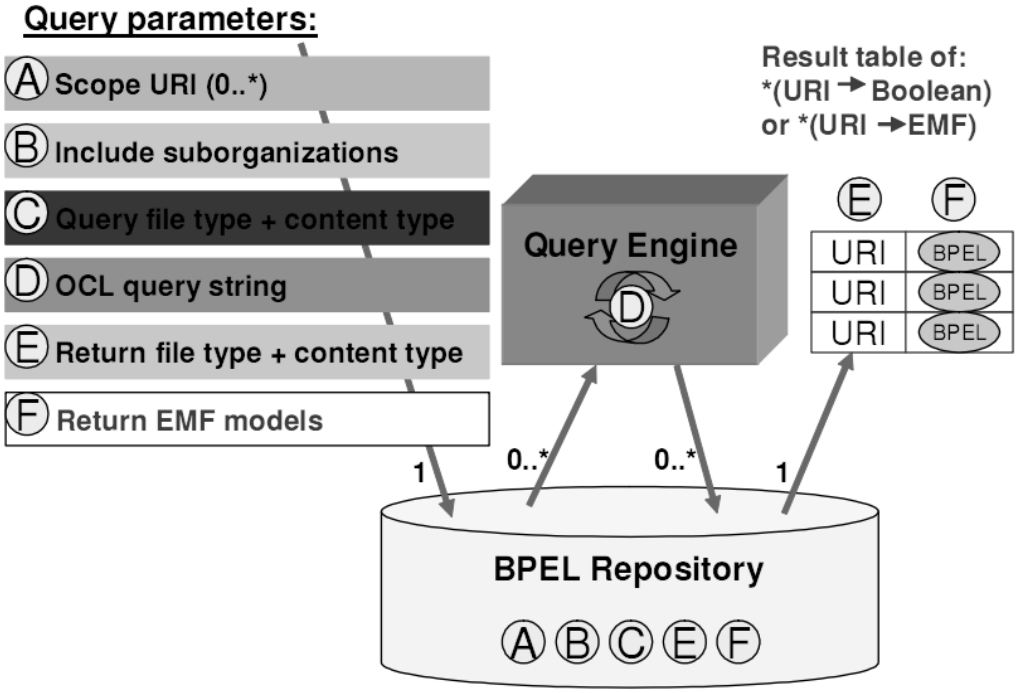


Figura A 28. Mecanismo de consulta del Repositorio BPEL

Hay seis parámetros de búsqueda. El parámetro (D) *OCL Query string*, es pasado al motor de consulta, los otros son usados dentro del repositorio. Los parámetros E y F determinan los tipos de salida en la tabla de resultados de la consulta.

B.2 Repositorio de Dispositivos

En vista de la necesidad de recuperar servicios relevantes para el usuario y que además puedan ser consumidos por cualquier dispositivo móvil, se ha creado el concepto de contexto de entrega. La aparición del contexto de entrega se hizo necesaria debido al creciente número de dispositivos que ingresan a Internet.

Algunas de las características que pueden definirse al interior de un contexto de entrega son las siguientes:

- *Interacción*: Diferentes formas de salida y entrada de información y sus parámetros.
- *Capacidades del Agente de Usuario*: Capacidades de procesamiento y soporte de distintos formatos.
- *Conexión*: Ancho de banda, latencia, Redes y protocolos, información asociada con los operadores de telecomunicaciones.
- *Localización*: Coordenadas geográficas, cercanía a otras fuentes y hora.
- *Local*: Lenguaje local y hora local.
- *Entorno*: Temperatura, iluminación y ruido.
- *Nivel de discurso*: Nivel de literatura y detalle del texto.
- *Confianza*: Privacidad y seguridad, restricciones de contenido.

Por otro lado, es necesario definir algunos conceptos relacionados con el procesamiento y el transporte del contexto de entrega:

a. *Estructura de datos y vocabulario para intercambio de contextos de entregas*: actualmente la recomendación para presentar la información es CCPP(Composite Capabilities/Preferences Profile).
b. Protocolo para la transmisión de *los contextos de entrega*: Aparte de usar las cabeceras del HTTP, la W3C ha implementado el CCPP exchange protocol basado en HTTP Extension Framework.
c. *Modelo de procesamiento para manejar la información del contexto de entrega*: Se relaciona muy cercanamente con el protocolo, y define reglas de generación e interpretación de las características, así como de actualización de estas.

La gestión del contexto para el presente proyecto de grado, además de lo expuesto, debe considerar dos variables: Una de ellas se introdujo en la sección 4.1. la cual considera los requerimientos de contexto del servicio, manejada por el repositorio de servicios. La segunda son las descripciones de las capacidades de los dispositivos que solicitan los servicios, denominada: *restricciones del contexto del solicitante*, que son las características de los dispositivos clientes tales como: capacidad de procesamiento, modalidades de presentación, interfaces de entrada, conectividad, etc. Esta información es recuperada por medio de dos repositorios: UAProf y WURFL.

B.2.1 UAProf - User Agente Profile

El Perfil de Agente de Usuario es una especificación definida por la OMA(Open Mobile Alliance), la cual es una organización que se ocupa de la definición de diversas normas relacionadas con las comunicaciones móviles, entre ellas las normas WAP. Una de las principales actividades de UAProf es recoger características y preferencias de dispositivos inalámbricos. El objetivo de esta especificación es dar solución a uno de los principales inconvenientes en el desarrollo de aplicaciones web para entornos multi-dispositivo, dependiendo de las capacidades del cliente móvil, se pueden ofrecer determinados servicios considerando sus restricciones.

UAProf fue implementado con el fin de que los fabricantes de dispositivos móviles pudieran describir y publicar las características técnicas y funcionalidades de sus dispositivos mediante el esquema XML y CC/PP, que es un sistema desarrollado por el W3C(World Wide Web Consortium). CC/PP es un marco más amplio que permite expresar no sólo la funcionalidad de los dispositivos móviles sino también las preferencias del usuario. De hecho, UAProf es la primera implementación de la norma CC/PP. Un gran número de fabricantes acogen a esta especificación para publicar sus productos. Actualmente, los fabricantes publican en su web los ficheros XML que describen las capacidades de sus dispositivos. Sin embargo, existe un problema con dichas publicaciones, y es que no hay homogeneidad en el uso de las reglas de CC/PP, de modo que los fabricantes utilizan términos diferentes para referirse al mismo atributo o funcionalidad. Además no están obligados a especificar todas las características del dispositivo.

El funcionamiento de UAProf consiste en enviar una URL por medio de las cabeceras HTTP, esta URL contiene las capacidades del dispositivo, descritas según el estándar CC/PP. Algunos problemas de este sistema son:

- No todos los dispositivos tienen UAProf
- Puede causar retardos en la navegación
- No existe un estándar para los datos en cada UAProf.
- Las cabeceras UAProf pueden planearse erróneamente.
- Algunos fabricantes crean estos perfiles para un grupo de dispositivos y no para cada producto en particular (Forum).

El esquema WAP UAProf consiste en la descripción de los siguientes componentes:

1. HadwarePlatform: Una colección de propiedades que describen apropiadamente el dispositivo terminal. Esto incluye, el tipo de dispositivo, numero del modelo, tamaño del display, métodos de entrada y salida de datos, etc.
2. SoftwarePlatform: Una colección de atributos relacionado con el entorno de operación del dispositivo. los atributos proveen información del sistema operativo usado, codificadores de video y audio soportados por el equipo y preferencias del lenguaje del usuario.
3. Browser UA: Un grupo de atributos para describir el browser HTML.
4. NetworkCharacteristics: Información acerca de la infraestructura de red y el entorno en el cual la información transita. los atributos son expresados en términos del ancho de banda de la red y la accesibilidad del dispositivo.
5. WapCharacteristics: Un grupo de atributos pertenecientes a las capacidades WAP soportadas en el dispositivo. Esto incluye las capacidades del Browser WML, Wireless Telephony Aplication, etc.
6. PushCaracteristics: Un grupo de características a las capacidades Push soportadas por el equipo. Esto incluye características de los MIME Types soportados por el equipo, el tamaño máximo de un mensaje push enviado del dispositivo, el tamaño del "buffer" de mensajes push que tiene el equipo, etc.

B.2.1.1 Uso del Vocabulario UAProf en CCPP

El siguiente es un ejemplo del uso del vocabulario UAProf en un perfil CCPP; se ha identificado mediante una marcación en llaves los componentes del esquema UAProf que están presentes en el ejemplo.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-0010430#">
 <rdf:Description rdf:ID="MyDeviceProfile">
 <prf:component>
 <rdf:Description rdf:ID="HardwarePlatform">
 <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010430#HardwarePlatform"/>
 <prf:BluetoothProfile>
 <rdf:Bag>
 <rdf:li>headset</rdf:li>
 </rdf:Bag>
 </prf:BluetoothProfile>
 <prf:ScreenSize>121x87</prf:ScreenSize>
 <prf:Model>R999</prf:Model>
 <prf:InputCharSet>
 <rdf:Bag>
 <rdf:li>ISO-8859-1</rdf:li>
```

```
 </rdf:Bag>
 </prf:InputCharSet>
 <prf:ScreenSizeChar>15x6</prf:ScreenSizeChar>
 <prf:BitsPerPixel>2</prf:BitsPerPixel>
 <prf:ColorCapable>No</prf:ColorCapable>
 <prf:TextInputCapable>Yes</prf:TextInputCapable>
 <prf:ImageCapable>Yes</prf:ImageCapable>
 <prf:Keyboard>PhoneKeypad</prf:Keyboard>
 <prf:NumberOfSoftKeys>0</prf:NumberOfSoftKeys>
 <prf:Vendor>myprofileprovider</prf:Vendor>
 <prf:OutputCharSet>
 <rdf:Bag>
 <rdf:li>ISO-8859-1</rdf:li>
 </rdf:Bag>
 </prf:OutputCharSet>
 <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
 <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
 </rdf:Description>
 </prf:component>
 <prf:component>
       <rdf:Description rdf:ID="SoftwarePlatform">
 <rdf:type
rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-
20010430#SoftwarePlatform"/>
 <prf:AcceptDownloadableSoftware>No</prf:AcceptDownloadableSoftware>
 </rdf:Description>
 </prf:component>
 <prf:component>
 <rdf:Description rdf:ID="NetworkCharacteristics">
 <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF
/ccppschema-20010430#NetworkCharacteristics"/>
 <prf:SecuritySupport>
 <rdf:Bag>
 <rdf:li>WTLS-1</rdf:li>
 </rdf:Bag>
 </prf:SecuritySupport>
 <prf:SupportedBearers>
 <rdf:Bag>
 <rdf:li>TwoWaySMS</rdf:li>
 </rdf:Bag>
 </prf:SupportedBearers>
  <prf:SupportedBluetoothVersion>1.1</prf:SupportedBluetoothVersion>
 </rdf:Description>
 </prf:component>
 <prf:component>
 <rdf:Description rdf:ID="BrowserUA">
 <rdf:type rdf:resource="http://www.wapforum.org/profiles/
UAPROF/ccppschema-20010430#BrowserUA"/>
 <prf:BrowserName>Ericsson</prf:BrowserName>
 <prf:CcppAccept>
 <rdf:Bag>
 <rdf:li>application/vnd.wap.wmlc</rdf:li>
 <rdf:li>text/x-vCard</rdf:li>
 <rdf:li>image/gif</rdf:li>
 </rdf:Bag>
 </prf:CcppAccept>
 <prf:CcppAccept-Charset>
 <rdf:Bag>
 <rdf:li>US-ASCII</rdf:li>
 </rdf:Bag>
 </prf:CcppAccept-Charset>
 <prf:CcppAccept-Encoding>
```

```
 <rdf:Bag>
 <rdf:li>base64</rdf:li>
 </rdf:Bag>
 </prf:CcppAccept-Encoding>
 <prf:FramesCapable>No</prf:FramesCapable>
 <prf:TablesCapable>Yes</prf:TablesCapable>
 </rdf:Description>
 </prf:component>
 <prf:component>
 <rdf:Description rdf:ID="WapCharacteristics">
 <rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF
/ccppschema-20010430#WapCharacteristics"/>
 <prf:WapDeviceClass>C</prf:WapDeviceClass>
 <prf:WapVersion>2.0</prf:WapVersion>
 <prf:WmlVersion>
 <rdf:Bag>
 <rdf:li>2.0</rdf:li>
 </rdf:Bag>
 </prf:WmlVersion>
 <prf:WmlDeckSize>3000</prf:WmlDeckSize>
 <prf:WmlScriptVersion>
 <rdf:Bag>
 <rdf:li>1.2.1</rdf:li>
 </rdf:Bag>
 </prf:WmlScriptVersion>
 <prf:WmlScriptLibraries>
 <rdf:Bag>
 <rdf:li>Lang</rdf:li>
 </rdf:Bag>
 </prf:WmlScriptLibraries>
 <prf:WtaiLibraries>
 <rdf:Bag>
 <rdf:li>WTA.Public.makeCall</rdf:li>
 </rdf:Bag>
 </prf:WtaiLibraries>
 </rdf:Description>
 </prf:component>
 </rdf:Description>
</rdf:RDF>
```

UAProf es una de las fuentes de información de WURFL, sistema que se estudiará a continuación.

B.2.2 WURFL - Wireless Universal Resource File

El Archivo universal de recursos inalámbricos es creado con el fin de dar solución a los diferentes inconvenientes detectados en UAProf, Haciendo que las características de los dispositivos inalámbricos estén en una única estructura. Las capacidades de los dispositivos encontradas en WURFL son proporcionadas a partir de las características publicadas por los fabricantes en UAProf y las contribuciones de desarrolladores y empresas de todo el mundo.

WURFL Es una iniciativa que gestiona un archivo XML de configuración el cual contiene información de las capacidades y características de un gran número de dispositivos móviles; frecuentemente, estas no son consignadas por los fabricantes sino por los desarrolladores que se ven en la necesidad de recopilarlos. El desarrollo de este documento XML es liderado desde sus inicios por Luca Passani y Andrea Trasatt

quienes a través de una gran comunidad de desarrolladores en todo el mundo tratan de mantener el repositorio lo más actualizado posible. Como parte de este trabajo, Lucas Passani creó WALL(Wireless Abstraction Library), una librería que permite diseñar páginas Web que dependiendo de las capacidades del dispositivo entrega el contenido en formato WML(Wireless Markup Language), C-HTML(Compact HyperText Markup Language), y XHTML(eXtensible HyperText Markup Language) Mobile Profile (Profile., 2010) El proyecto empezó en enero de 2002 y desde entonces se han reunido datos de más de 7.000 dispositivos.

ANEXO C

MANUAL DE INSTALACIÓN Y DE USUARIO DEL REPOSITORIO DE PROCESOS BPEL

C.1 Instrucciones de Instalación

Hay dos formas diferentes para instalar y usar el Repositorio BPEL:

1. Se puede utilizar como software de código abierto (OSS), cuando se instala en la plataforma Eclipse. siga las instrucciones de instalación en la Sección XX.

2. La otra opción es usar el repositorio junto con el producto IBM Rational Software Architect. Entonces puede utilizar el motor de IBM OCL, que forma parte del producto. El beneficio de la instalación del repositorio en la arquitectura de IBM® Rational® Software Architect 6,0 (RSA) en lugar de la plataforma Eclipse es que el Repositorio puede utilizar el motor de consulta IBM OCL, que forma parte de RSA. El motor de IBM OCL consulta no está disponible como software de código abierto. Es por ellos que esta parte no será tratada en el presente documento.

C.1.1 Instalación en Eclipse

Esta sección le guiará a través de la instalación del Repositorio en la plataforma Eclipse:

1. Como requisito previo, debe tener **Java 2 SDK 1.4.2** o **JDK 5.0** instalado en su equipo. Si usted aún no lo tiene, puede encontrarlos en:

   a. http://www-128.ibm.com/developerworks/java/jdk/
   b. http://java.sun.com/j2se/1.4.2/index.jsp

2. Instalar **Eclipse SDK 3.0.2** (http://www.eclipse.org/eclipse/). El Repositorio puede trabajar con todas las versiones 3.x de Eclipse (Hay pruebas exitosas con otras versiones, como el 3,0 y 3,1). Sin embargo, Eclipse 3.0.2 es la versión recomendada.

   a. Descargar Eclipse SDK (por ejemplo, la versión **3.0.2**) Desde:
      http://www.eclipse.org/downloads/index.php
   b. Extraiga el archivo zip descargado (por ejemplo, **eclipse-SDK-3.0.2-win32.zip**) para su sistema de archivos. Usted puede libremente seleccionar la ubicación, que se convertirá en el Eclipse carpeta de instalación (puede usar por ejemplo, **C:\eclipse\SDK-3.0.2**).

3. Instale **EMF/XSD/SDO** con la versión correcta para Eclipse 3.0.2, por ejemplo la **2.0.4**, (http://www.eclipse.org/emf/). Si está utilizando otras versiones de Eclipse SDK 3.x, seleccione la versión de EMF/XSD/SDO que sea compatible con ella.

   a. Download **EMF/XSD/SDO SDK** (e.g. version 2.0.4) from:
   http://download.eclipse.org/tools/emf/scripts/downloads.php

b. Extraiga el archivo zip descargado (por ejemplo, **emf-sdo-xsd-SDK-2.0.4.zip**) en la carpeta de instalación de Eclipse previamente seleccionada (por ejemplo,**C:\eclipse\SDK-3.0.2**).

4. Instale el plugin del Repository BPEL en la instalación de Eclipse:

   a. Descargue el archivo **BPEL-Repository-2.1.0.zip** de: http://www.alphaworks.ibm.com/tech/bpelrepository
   b. Extraiga el archivo **BPEL-Repository-2.1.0.zip** en algún lugar de su sistema (por ejemplo **C:\temp**).
   c. Busque el archivo **BPEL-Repository-2.1.0-for-Eclipse.zip** de la subcarpeta de instalación de los archivos (por ejemplo **C:\temp\archivos de instalación**).
   d. Extraiga el archivo **BPEL-Repository-2.1.0-for-Eclipse.zip** en una subcarpeta de tu carpeta de instalación de Eclipse (por ejemplo **C:\eclipse\SDK-3.0.2\eclipse**).

C.2 Comience a usar el Repositorio BPEL

Esta sección muestra los escenarios de uso más típicos del Repositorio BPEL con ejemplos. Los escenarios en éste capítulo se describen para la instalación del repositorio en la plataforma Eclipse, pero los pasos son casi idénticos para la instalación de IBM Rational Software Architect, porque el producto se basa en Eclipse.

C.2.1 Use el repositorio con el contenido de muestra

Siga las instrucciones para aprender a abrir el Repositorio con el contenido de muestra.

1. Copie el contenido de la muestra en su computadora.

   a. Busque el archivo **Sample-Contents.zip** en la subcarpeta **repository-contents** que está incluida en el archivo **BPEL-Repository-2.1.0.zip** descargado desde el sitio web de IBM alphaWorks en: http://www.alphaworks.ibm.com/tech/bpelrepository
   b. Extraiga el archivo **Sample-Contents.zip** en su sistema de archivos. Puede seleccionar la ubicación que desee (por ejemplo **C:Repositorio**).

2. Inicie Eclipse.

   a. Ejecute el archivo **eclipse.exe** en el directorio de instalación de Eclipse: por ejemplo,**C:\eclipse\SD-3.0.2\eclipse\eclipse.exe**
   b. Si lo desea, puede crear un acceso directo al archivo en el escritorio, y usar el acceso directo en su lugar.
   c. Establecer el Workbench por ejemplo, **C:\workspaces\repository\sample-contents.**
   d. Seleccione el icono en la esquina superior derecha (**Go to the workbench**).

3. Abra la perspectiva **BPEL Repository**. Adicionalmente, el **Organization Explorer** y la vista **Organization Contents** se abren en la perspectiva.

a. Seleccione **Window** → **Open Perspective** → **Other**… (Ver Figure A27) → **BPEL Repository** → **OK**. (Ver Figura A28).

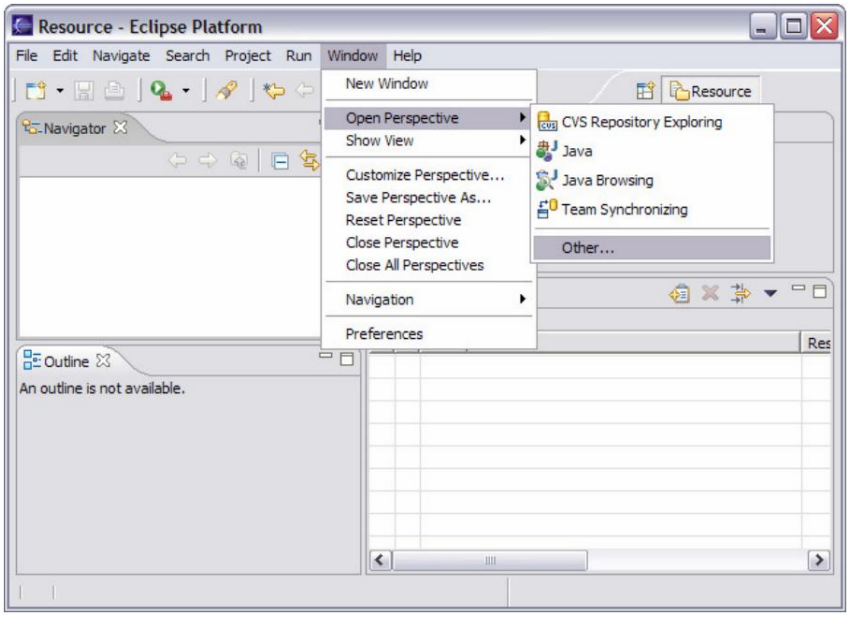


Figura A 29. Perspectiva del Repositorio BPEL.

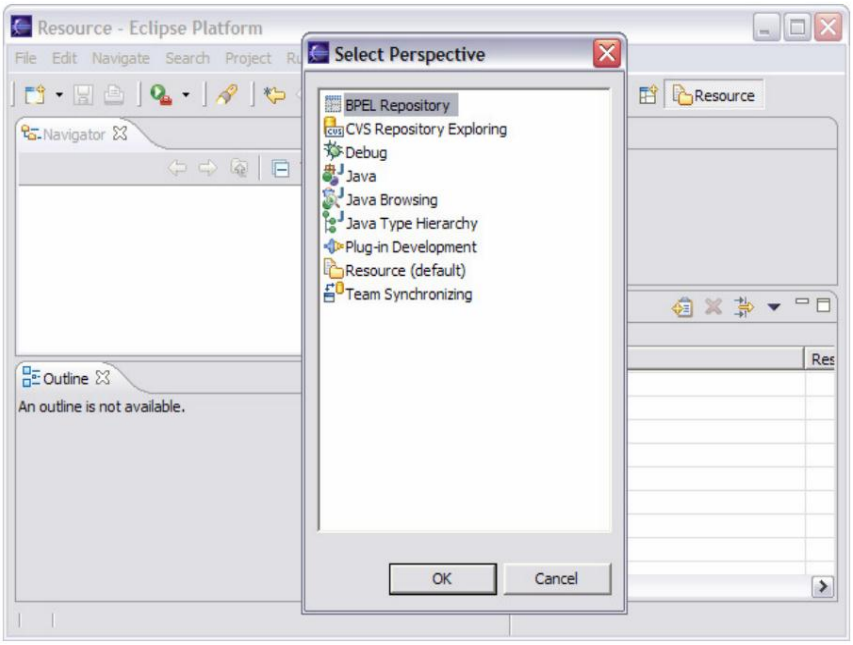


Figura A 30. Selección de la Perspectiva del Repositorio BPEL.

4. Click derecho en la vista vacía **Organization Explorer** → **Open Repository…** (ver Figura A29) → **Browse…** → Seleccione el subdirectorio **Sample-Contents** donde ha almacenado el contenido del **Sample-Contents.zip** (e.g. para el directorio

C:\Repositories\Sample-Contents, la URI es **file:/C:/Repositories/Sample-Contents**, ver Figura A30). Seleccione **Finish.**

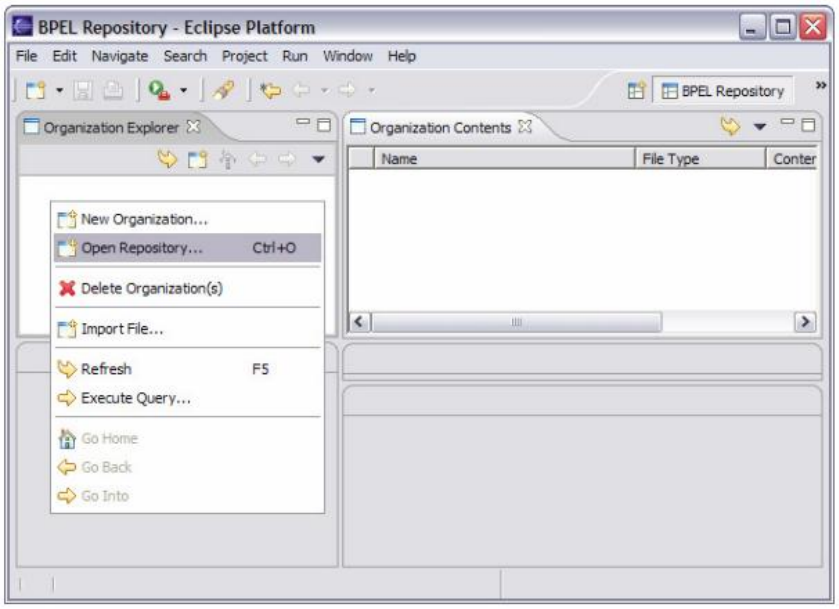


Figura A 31. Abrir el Repositorio BPEL.

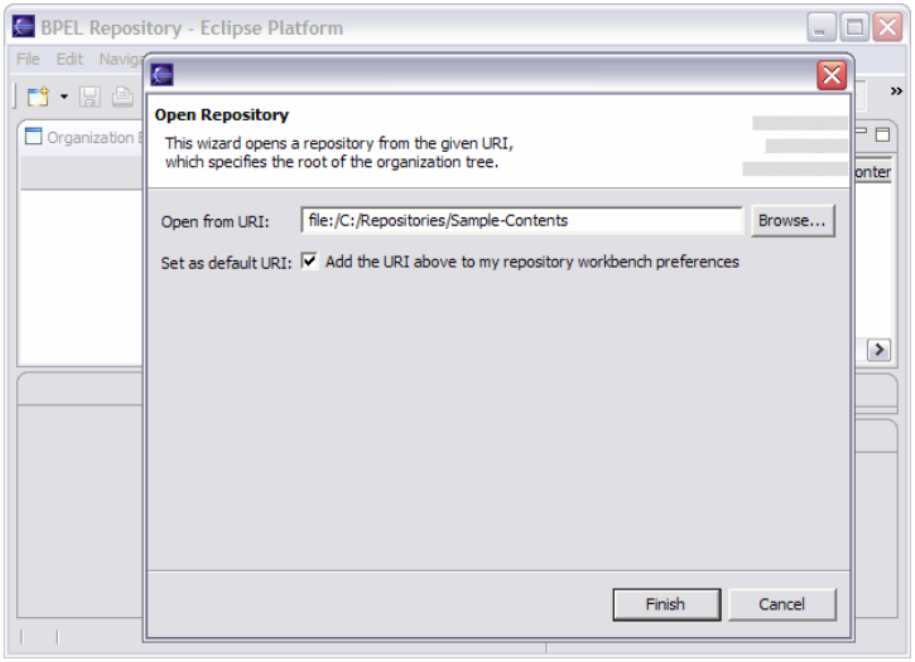


Figura A 32. Abrir el Repositorio desde una ubicación especifica.

C.2.1.1 Archivos de consultas en las Organizaciones

Usted puede seguir el sencillo ejemplo a continuación para saber cómo consultar el Repositorio:

1. Seleccione la organización **TravelReservation** en el árbol de la vista **Organization Explorer** vista y haga clic derecho en él. Seleccione **Execute Query** (Figura A31.)

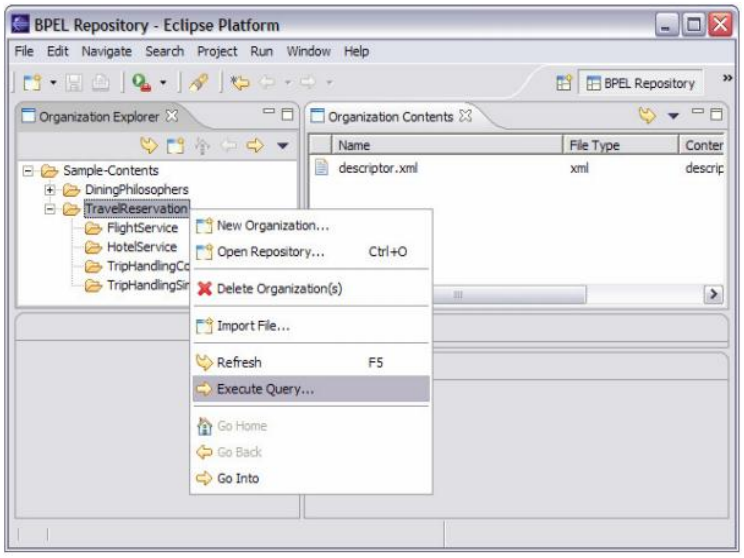


Figura A 33. Ejecutar una Consulta.

2. Cambiar los siguientes parámetros de consulta y dejar los valores por defecto en los otros parámetros, véase la Figura A32:

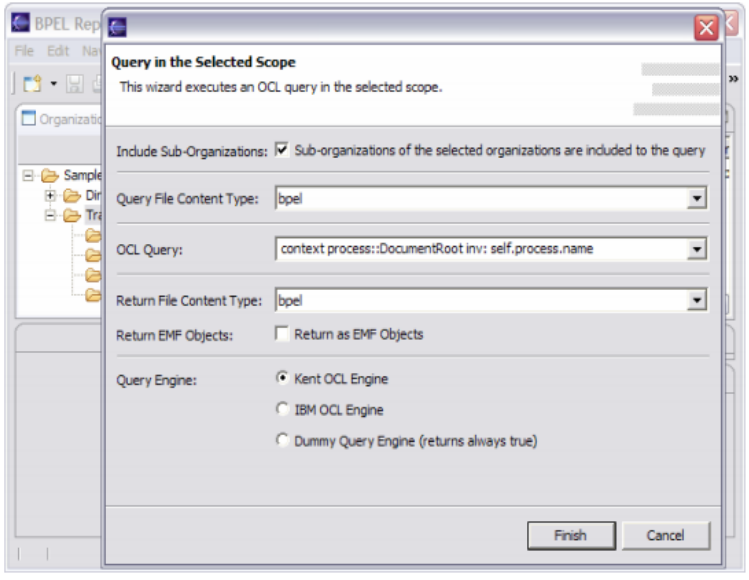


Figura A 34. Selección de Parámetros de Consulta.

a. Seleccione el tipo de archivo de consulta: **bpel**

b. Escriba una consulta OCL, por ejemplo, para encontrar los nombres de los procesos:
context process::DocumentRoot inv: self.process.name
c. Selecione el tipo de archivo de retorno: **bpel.**

3. Click **Finish**
4. Usted puede ver los resultados de la consulta en la vista de **Query Results**, vea la Figura A33. En el primera columna, está la URI para cada archivo retornado de cada archivo consultado. En la segunda columna, **Query Result**, los objetos EMF se muestran como una cadena. Y si no hay objetos EMF para mostrar, entonces el resultado booleano de cada archivo consultado es mostrado. Los resultados que de falso contenido son filtrados.

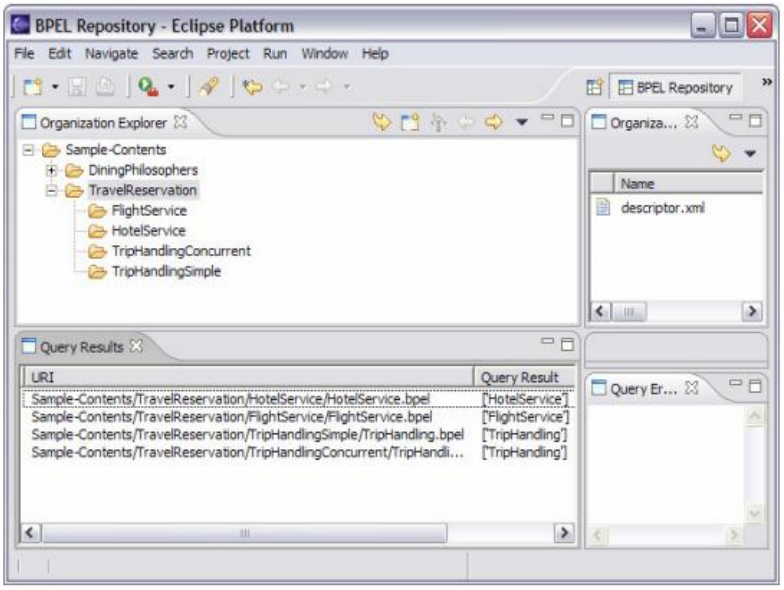


Figura A 35. Resultados de la Consulta.

C.2.1.2 Crear una nueva Organización

Los pasos para crear una organización son presentados a continuación:

1. Click derecho en la organización (directorio) en la cual desea crear la nueva organización (directorio). **Sample-Contents**, y seleccione **New Organization…,** ver Figure A34.

2. Escriba el nombre de la nueva organización, por ejemplo **NewOrganization…** Figura A35.

3. Selecione **Finish**

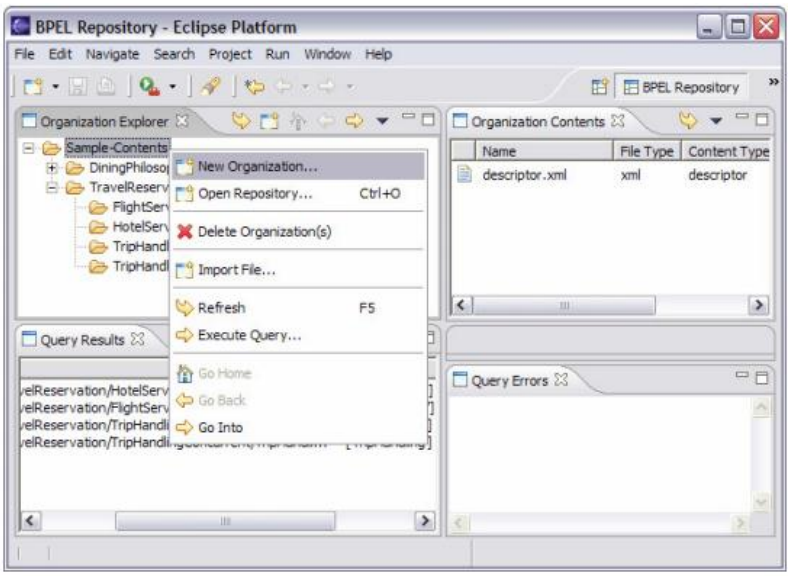


Figura A 36. Crear una nueva Organización

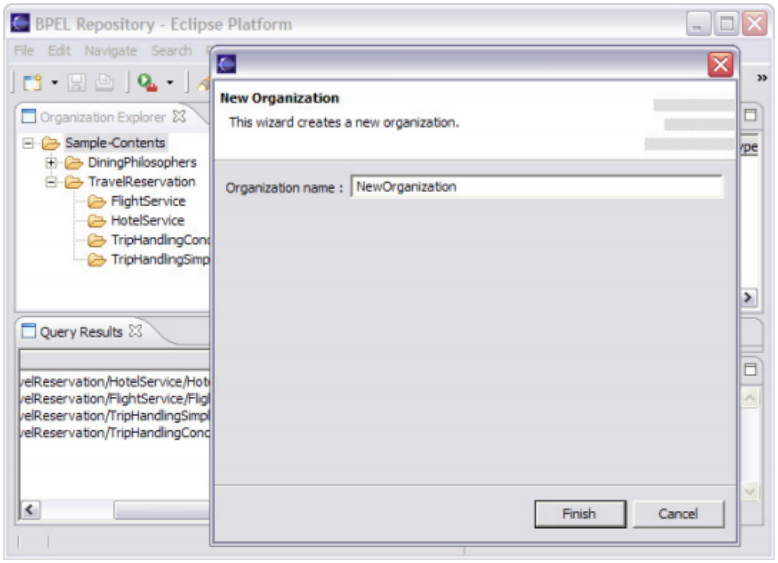


Figura A 37. Nombre de la nueva Organización.

C.2.1.3 Importar un Archivo

En esta sección se muestra como un archivo puede ser importada en el Repositorio.

1. Click derecho en la organización donde desea importar el archivo, por ejemplo: **NewOrganization**, y seleccione **Import File** (Figura A36).

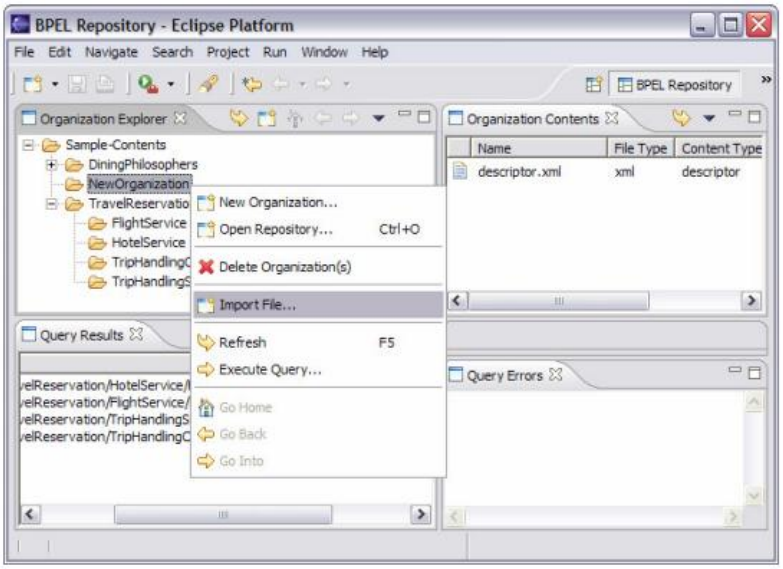


Figura A 38. Importar un Archivo.

2. Seleccione **Browse…** (ver Figura A37.) para encontrar el archivo que desee importar. Por ejemplo:*C:\Repositories\SampleContents\TravelReservation\FlightService/FlightService.bpel*

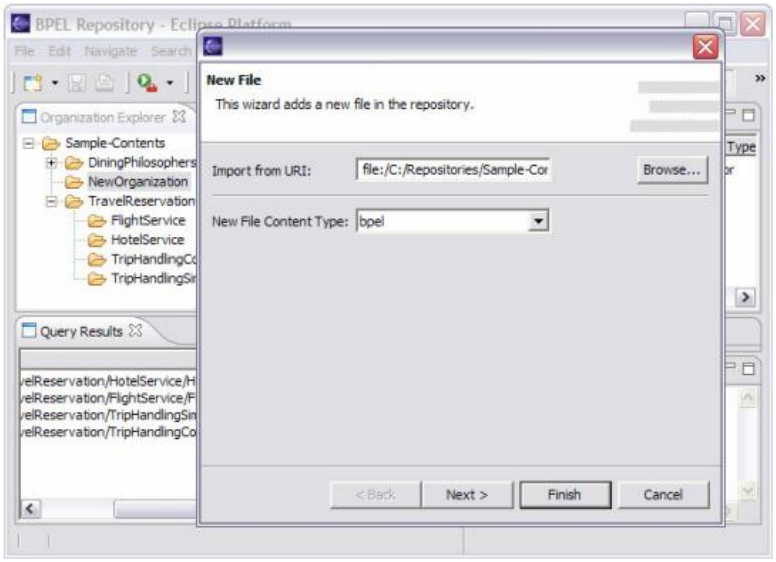


Figura A 39. Seleccionar la ubicación de donde el archivo es importado.

3. Seleccione el **New File Content Type** para el archivo.
4. **Next**
5. Cambie el nombre del archivo, si lo desea; por ejemplo, ***NewFlightService.bpel***. Preste atención a que el fichero termina con la extensión asociada al tipo de contenido seleccionado. También puede cambiar el tipo de contenido y ver qué tipo de archivo y extensión de archivo está relacionado a cada tipo de contenido. Véase el gráfico A38.
6. **Finish.**
7. Haga clic en la organización, donde se importó el archivo y seleccione Refresh. El archivo aparecerá en la vista de Organización del contenido, véase la Figura A39.

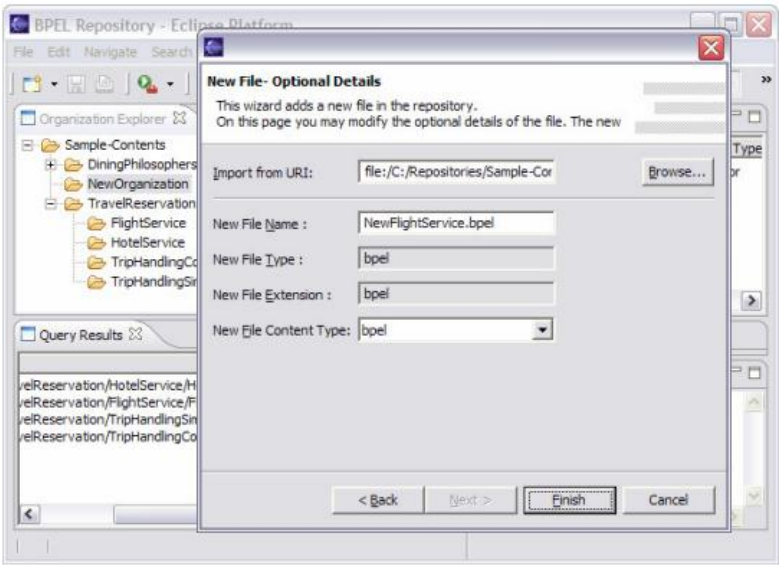


Figura A 40. Cambio de nombre del Archivo Importado.

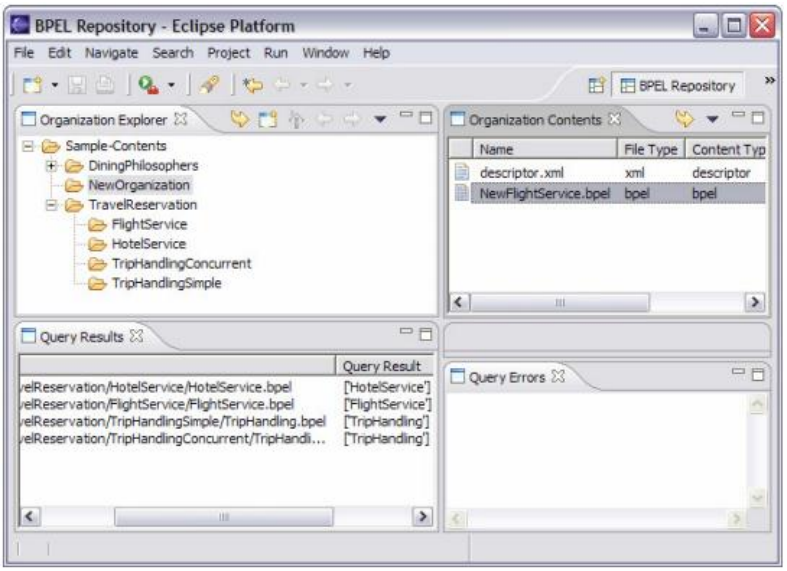


Figura A 41. Nuevo archivo Importado.

C.2.2 Extender el Repositorio con un nuevo EMF Model.

C.2.2.1 Cree su propio modelo EMF

El modelo EMF se puede crear a partir de tres modelos existentes:

1. Rational Rose model file
2. Esquema XML
3. Anotaciones Java.

Utilizaremos anotaciones Java para crear el modelo EMF para el ejemplo de muestra en los siguientes apartados. El valor por defecto de los modelos EMF para BPEL, WSDL y XSD, se pueden descargar junto con el repositorio, los cuales son creados a partir del estándar XML.

Esta sección le guiará a través de la creación del modelo de muestra EMF.

1. Importe el proyecto Java que desea utilizar como fuente para la generación de modelos EMF, por ejemplo, importe el **EMF-Sample-Code_1.0.0_source-code-project.zip** que es incluido en la subcarpeta **source-code-examples**, ejemplos del **BPEL-Repository-2.1.0.zip** que se puede descargar desde el sitio web de IBM alphaWorks. El proyecto importado se muestra en la Figura 6.1.

Figura A 42. Un proyecto Java importado que contienen anotaciones Java para la generación del modelo EMF.

   a. Extraiga el ***EMF-Sample-Code_1.0.0_source-code-project.zip*** en su carpeta W**orkspace.**
   b. Retorne a Eclipse y seleccione **File → Import… → Existing Project into Workspace → Next.**
   c. contenido del proyecto, use **Browser**... para localizar la carpeta extraída en el área de trabajo **Workspace**, por ejemplo, *C:\workspaces\repository\emf-example\EMF-Sample-Code.*
   d. **Finish.**

2. Haga clic en el paquete de Java (por ejemplo, *com.ibm.bpia.repository.extensions.emf-muestra*) del que desea generar el modelo EMF.

   a. Selecione **New → Other… → Eclipse Modeling Framework catalog → EMF Models → Next.**
   b. Introduzca o seleccione la carpeta principal, seleccione la carpeta en la que el código Java se encuentra, por ejemplo, *EMF-Sample-Code/com/ibm/bpia/repository/extensions/emf-muestra.*

c. Establecer ***File name*** como: *com.ibm.bpia.repository.extensions.emfsample.gen-modelo* (ver Figura 6.2).
d. **Next.**

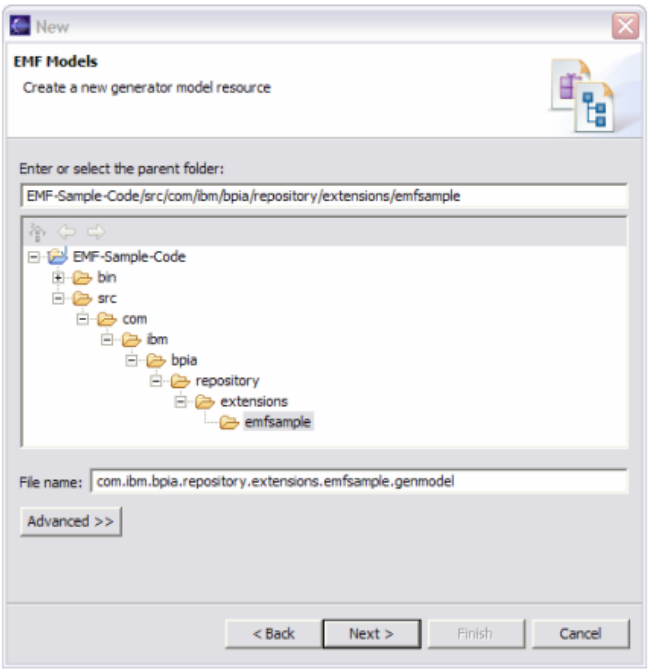


Figura A 43. Cambiar nombre del archivo genmodel.

3. Seleccione **Load from annotated Java y Next.**
4. Seleccione los paquetes que desee incluir en el modelo.

5. Cambiar el nombre de **Core Model Name** con el valor *emfsample.ecore* a *com.ibm.bpia.repository.extensions.emfsample.ecore* (Ver Figura 6.3). Seleccionar **Finish.**

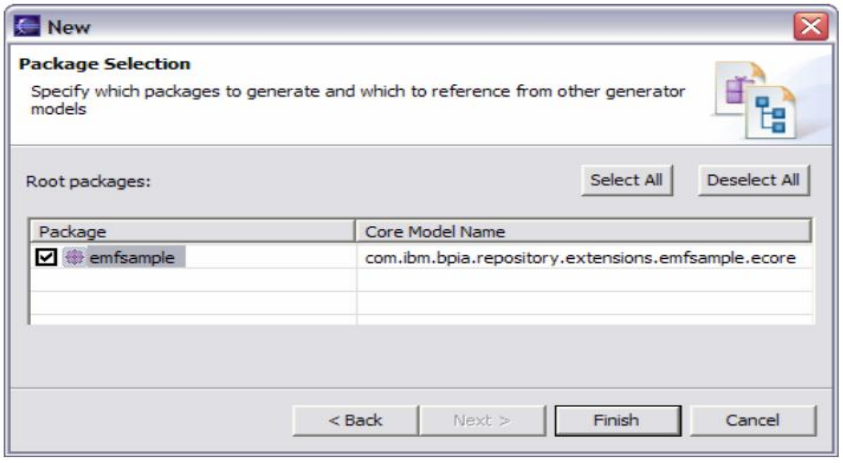


Figura A 44. Cambiar nombre al archivo ecore

6. Modificar las propiedades del archivo **ecore**
   a. Abrir el archivo *com.ibm.bpia.repository.extensions.emfsample.ecore*
   b. Click derecho en el paquete *emfsample* y seleccionar **Show Properties View.**
   c. Acortar el prefijo espacio de nombres a fin de que los archivos XML más fácil de leer, el cambio de prefijo para Ns: emfsample. Esto también hace que la línea aérea de contacto consultas más corto. Véase la figura A43.
   d. Cambiar el **Ns Prefix** a: *emfsample.*

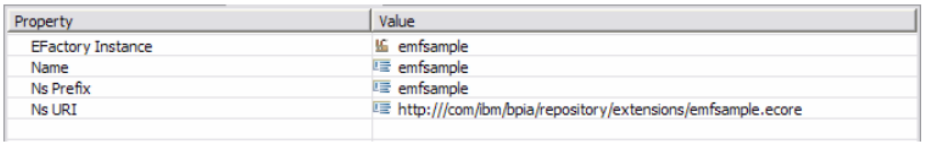


Figura A 45. Cambiar la propiedad Ns Prefix a: *emfsample*

   e. En la clase *Customer*, seleccione el atributo *address : Address* y establezca el valor de *Containment* a **True.** Ver Figura A44.

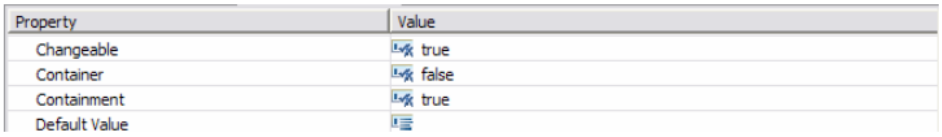
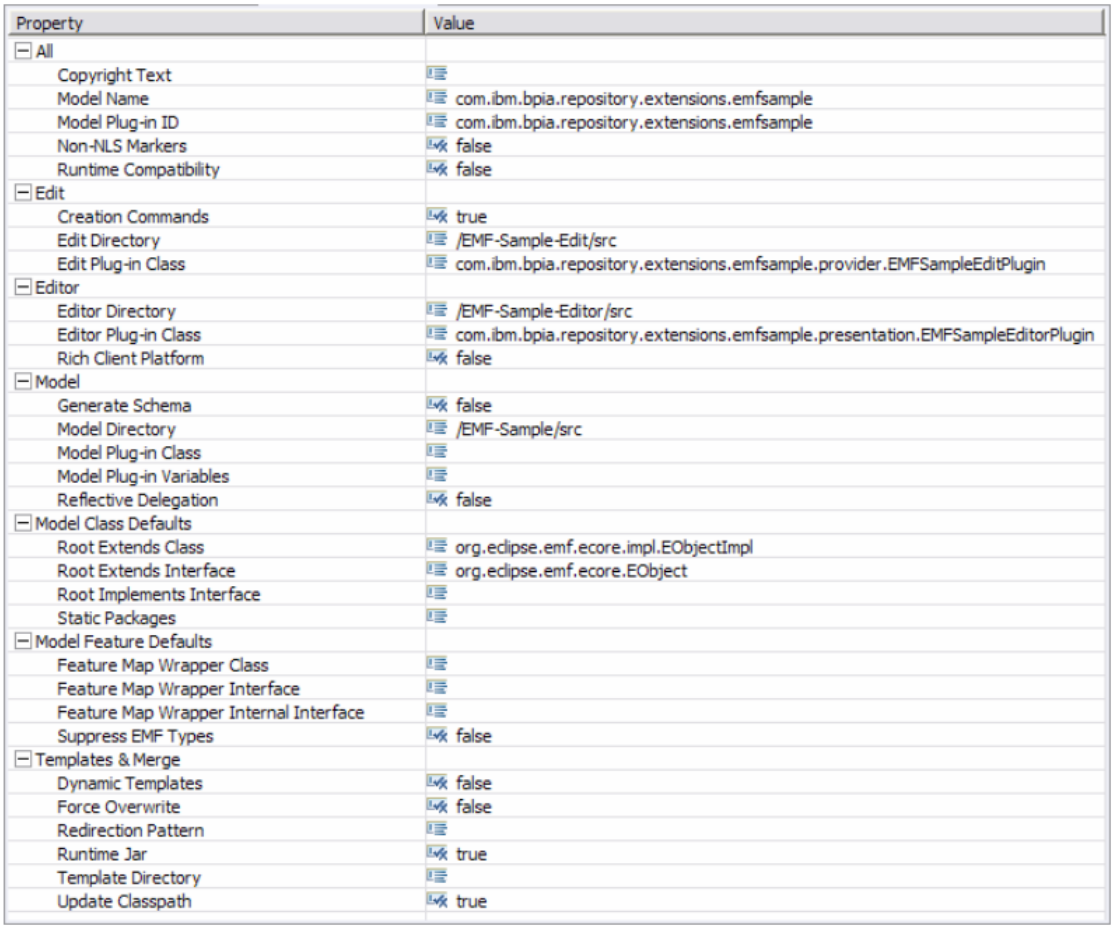


Figura A 46. Cambiar el valor de la propiedad *Containment* a True

7. Modificar las propiedades del archivo *genmodel.* Ver figura A45.

   a. Abrir el archive com.ibm.bpia.repository.extensions.emfsample.genmodel.
   b. Seleccionar el elemento raíz: *com.ibm.bpia.repository.extensions.emfsample*.
   c. Asignar a **Model Plug-in ID:** com.ibm.bpia.repository.extensions.emfsample.
   d. Asignar a **Edit Directory:** /EMF-Sample-Edit/src.
   e. Asignar a **Editor Directory:** /EMF-Sample-Editor/src.
   f. Asignar a **Edit Plug-in Class:** *com.ibm.bpia.repository.extensions.emf-sample.provider.EMFSampleEditPlugin.*
   g. Asignar a **Editor Plug-in Class**: *com.ibm.bpia.repository.extensions.emf-sample.presentation.EMFSampleEditorPlugin.*
   h. Asiganar a **Model Directory:** /EMF-Sample/src.

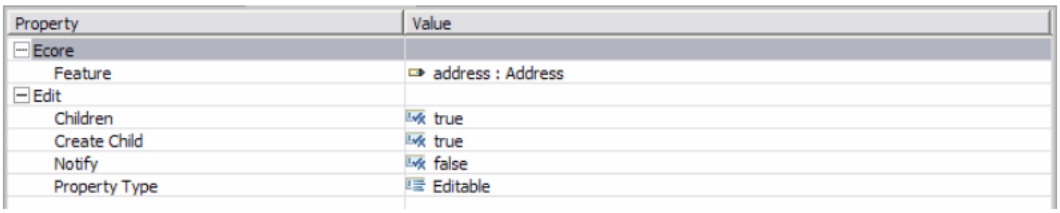


Property	Value
All	
Copyright Text	
Model Name	com.ibm.bpia.repository.extensions.emfsample
Model Plug-in ID	com.ibm.bpia.repository.extensions.emfsample
Non-NLS Markers	false
Runtime Compatibility	false
Edit	
Creation Commands	true
Edit Directory	/EMF-Sample-Edit/src
Edit Plug-in Class	com.ibm.bpia.repository.extensions.emfsample.provider.EMFSampleEditPlugin
Editor	
Editor Directory	/EMF-Sample-Editor/src
Editor Plug-in Class	com.ibm.bpia.repository.extensions.emfsample.presentation.EMFSampleEditorPlugin
Rich Client Platform	false
Model	
Generate Schema	false
Model Directory	/EMF-Sample/src
Model Plug-in Class	
Model Plug-in Variables	
Reflective Delegation	false
Model Class Defaults	
Root Extends Class	org.eclipse.emf.ecore.impl.EObjectImpl
Root Extends Interface	org.eclipse.emf.ecore.EObject
Root Implements Interface	
Static Packages	
Model Feature Defaults	
Feature Map Wrapper Class	
Feature Map Wrapper Interface	
Feature Map Wrapper Internal Interface	
Suppress EMF Types	false
Templates & Merge	
Dynamic Templates	false
Force Overwrite	false
Redirection Pattern	
Runtime Jar	true
Template Directory	
Update Classpath	true

Figura A 47. Propiedades del archivo *genmodel*

8. Seleccione el atributo **adres:Address** de la clase **Customer** y permita crearlo con el editor (ver Figura A):

   a. Asignar a Children: True
   b. Asignar a Create Child: True



Property	Value
Ecore	
Feature	address : Address
Edit	
Children	true
Create Child	true
Notify	false
Property Type	Editable

Figura A 48. Propiedades del atributo *address*

9. Seleccione **File → Save All** para guarder los cambios en los archivos **ecore** y **genmodel**.

Para el caso del presente proyecto de grado se definió un nuevo modelo esquema EMF denominado *features.context* siguiendo los pasos anteriormente descritos.

Para ello fue necesario extender el Repositorio para que soporte varios modelos EMFs. Gracias a la flexibilidad del Repositorio BPEL es posible extender su funcionalidad a multiples modelos EMF implementando solo una simple clase para multiples modelos EMF.

Esta clase debe extender la clase **AbstractEMFExtensionCollection**, en lugar de la clase **EMFExtensionCollection**, la cual es usada para una sola extensión EMF. Un ejemplo de la implementación de esta clase es presentado en el Listado A1. Esta clase se utiliza para registrar los modelos EMF para **BPEL**, **WSDL** y archivos **XSD** con el **BPEL Repository Default Extensions plug-in**.

```
package com.ibm.bpia.repository.extensions.defaultemf;

import org.w3._2001.xml.schema.SchemaPackage;
import org.w3._2001.xml.schema.util.SchemaResourceFactoryImpl;
import org.xmlsoap.schemas.ws._2003._03.business.process.ProcessPackage;
import
org.xmlsoap.schemas.ws._2003._03.business.process.util.ProcessResourceFactoryI
mpl;
import org.xmlsoap.schemas.wsdl.WsdlPackage;
import org.xmlsoap.schemas.wsdl.util.WsdlResourceFactoryImpl;

import com.ibm.bpia.repository.Constants;
import com.ibm.bpia.repository.FileType;
import com.ibm.bpia.repository.extensions.AbstractEMFExtensionCollection;
import com.ibm.bpia.repository.extensions.GenericEMFExtension;
/**
 * Registers the default EMF models for BPEL, WSDL and XSD files with their
 * content types and other attributes (file types and filename extensions).
 * <p>
 * © Copyright IBM Corporation 2006. All Rights Reserved.
 * </p>
 * @author Jussi Vanhatalo
 */
public class DefaultEMFExtensionCollection extends
        AbstractEMFExtensionCollection {

    /**
     * Filename extension for BPEL files.
     */
    public static final String EXTENSION_BPEL = Constants.BPEL;

    /**
     * The filename extension for WSDL files.
     */
    public static final String EXTENSION_WSDL = Constants.WSDL;

    /**
     * The filename extension for XSD files.
     */
    public static final String EXTENSION_XSD = Constants.XSD;

    /**
     * The content type of BPEL files.
     */
    public static final String CONTENT_TYPE_BPEL = Constants.BPEL;
```

```
    /**
     * An alternative content type of BPEL files.
     */
    public static final String CONTENT_TYPE_BPEL_OPTIONAL = "";

    /**
     * The content type of WSDL files containing a public interface.
     */
    public static final String CONTENT_TYPE_WSDL_PUBLIC_INTERFACE =
        Constants.PUBLIC_INTERFACE;

    /**
     * The content type of WSDL files containing a partner links.
     */
    public static final String CONTENT_TYPE_WSDL_PARTNER_LINKS =
        Constants.PARTNER_LINKS;

    /**
     * The content type of XSD files.
     */
    public static final String CONTENT_TYPE_XSD = "xml-schema";

    /**
     * The default constructor.
     */
    public DefaultEMFExtensionCollection() {
        super();
    }

    /**
     * Registers the EMF models with the content types and the other
     * attributes (file types and filename extensions).
     */
public void registerEMFExtensionCollection() {
        GenericEMFExtension.createRegisteredEMFExtension(EXTENSION_BPEL,
                CONTENT_TYPE_BPEL, new ProcessResourceFactoryImpl(),
                ProcessPackage.eINSTANCE, FileType.BPEL_LITERAL);
        GenericEMFExtension.createRegisteredEMFExtension(EXTENSION_BPEL,
                CONTENT_TYPE_BPEL_OPTIONAL, new ProcessResourceFactoryImpl(),
                ProcessPackage.eINSTANCE, FileType.BPEL_LITERAL);

        GenericEMFExtension.createRegisteredEMFExtension(EXTENSION_WSDL,
                CONTENT_TYPE_WSDL_PARTNER_LINKS, new
WsdlResourceFactoryImpl(),
                WsdlPackage.eINSTANCE, FileType.WSDL_LITERAL);
        GenericEMFExtension.createRegisteredEMFExtension(EXTENSION_WSDL,
                CONTENT_TYPE_WSDL_PUBLIC_INTERFACE,
                new WsdlResourceFactoryImpl(), WsdlPackage.eINSTANCE,
                FileType.WSDL_LITERAL);

        GenericEMFExtension.createRegisteredEMFExtension(EXTENSION_XSD,
                CONTENT_TYPE_XSD, new SchemaResourceFactoryImpl(),
                SchemaPackage.eINSTANCE, FileType.XSD_LITERAL);
    }
}
```

Listado A1. Código fuente para la clase SampleEMFExtension

En el método **registerEMFExtensionCollection**, una clase **GenericEMFExtensiondebe** ser creados para cada tipo de contenido asociado a un modelo EMF. Cuando el **GenericEMFExtension** clase se crea con el método **createRegisterdEMFExtension**, esta es automáticamente registrada como una extensión EMF. Extender la clase **AbstractEMFExtensionCollection** es una forma alternativa para extender el repositorio con varios modelos EMF con el mínimo esfuerzo.

Para que la clase **DefaultEMFExtensionCollection** pueda registrar y por ende soportar el metadato de contexto definido, es necesario adicionar las siguientes líneas.

```
/**
* The content type of new metadata for Context.
*/
public static final String CONTENT_TYPE_EMF_SAMPLE = "EMF-Context";
public static final String EXTENSION_EMF_SAMPLE = "context";
public static final FileType FILE_TYPE_EMF_SAMPLE = FileType.METADATA_LITERAL;
```

Estos atributos permiten registrar como una extensión EMF el en nuevo modelo: `context`.

ANEXO D

TABLAS DE RESULTADOS DE LAS PRUEBAS DE CALIDAD Y RENDIMIENTO DEL SISTEMA

D.1 Transformación de los procesos BPEL almacenados en el Repositorio de Servicios

D.1.1 Prueba de rendimiento TR1

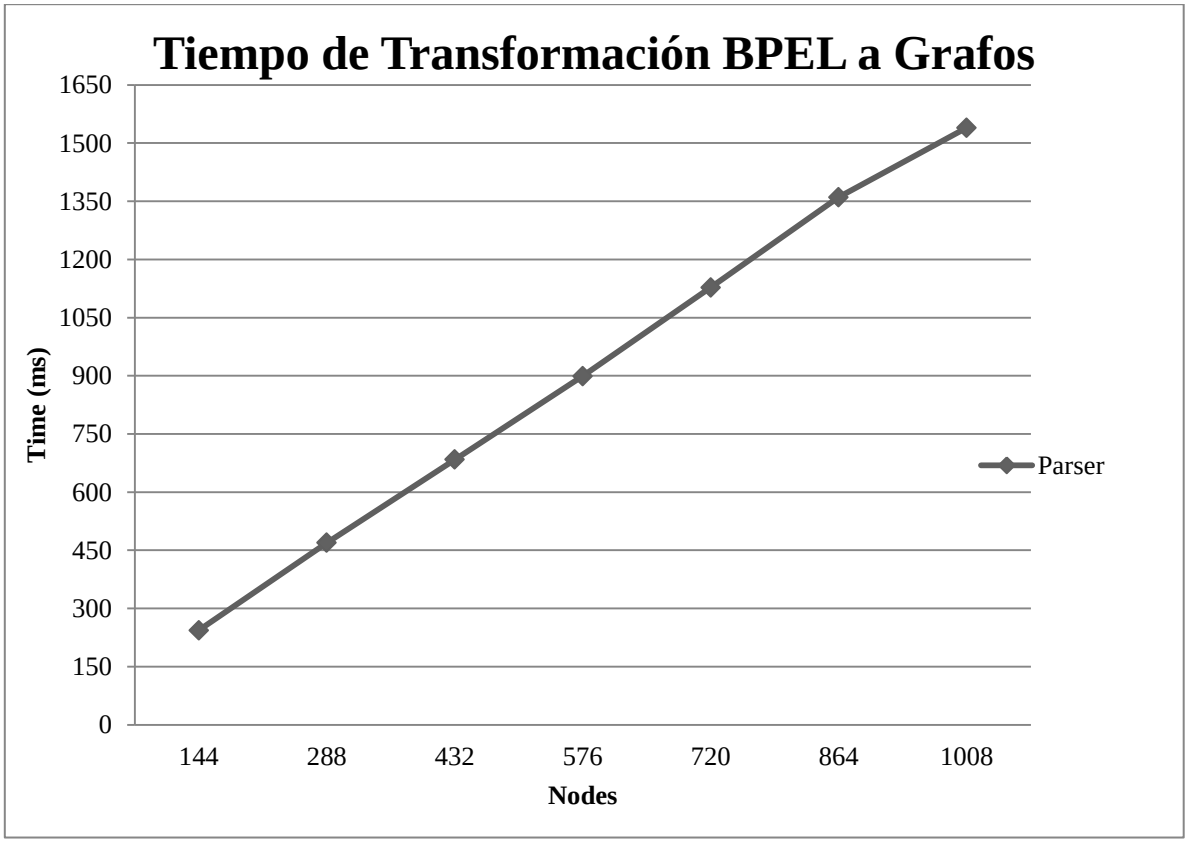


Figura A 49. Rendimiento del proceso de Transformación de los documentos BPEL del Repositorio de Servicios en Grafos, además de obtener, clasificar y organizar los nodos por tipo de actividad.

Nodes	Parser Time (millisecond)					Total
	Dominio1	Dominio2	Dominio3	Dominio4	Dominio5	
144	272	267,4	227,8	224,6	227,8	**243,92**
288	516,8	536,6	439,8	431,4	424,2	**469,76**
432	734	742,4	655,6	633,4	654,8	**684,04**
576	977,4	929,6	842,4	867,4	879,6	**899,28**
720	1221,2	1145	1080	1092	1101,2	**1127,88**
864	1492	1366,6	1347,8	1332,2	1263,8	**1360,48**
1008	1656,2	1528,8	1497,4	1538,4	1476,8	**1539,52**

D.2 Recuperación de Actividades Básicas BPEL (ServiceMatch)

D.2.1 Prueba de Calidad de los Resultados EC1

Para el análisis de los resultados se plantean tres escenarios:

Caso 1: evaluación de las medidas de desempeño comparando las actividades de entrada contra las actividades del repositorio que pertenecen al mismo dominio.

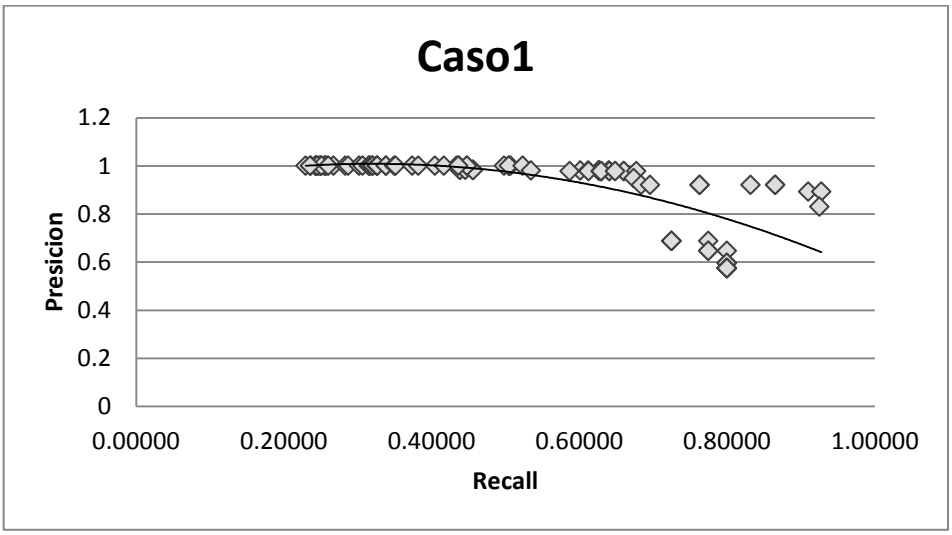
Caso 2: evaluación comparando las actividades de entrada contra las actividades almacenadas que pertenecen a un dominio diferente.

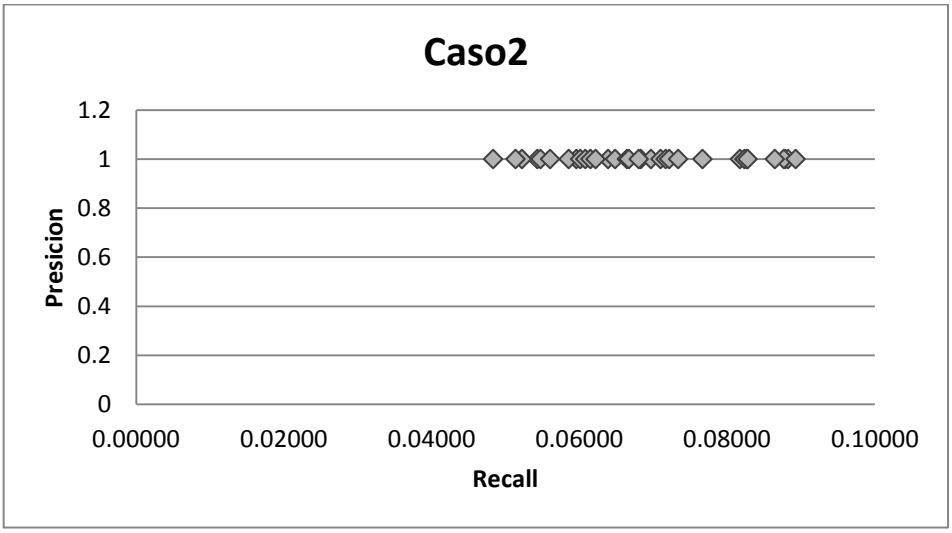
Caso 3: evaluación comparando las actividades de consulta contra todas las actividades contenidas en el repositorio.

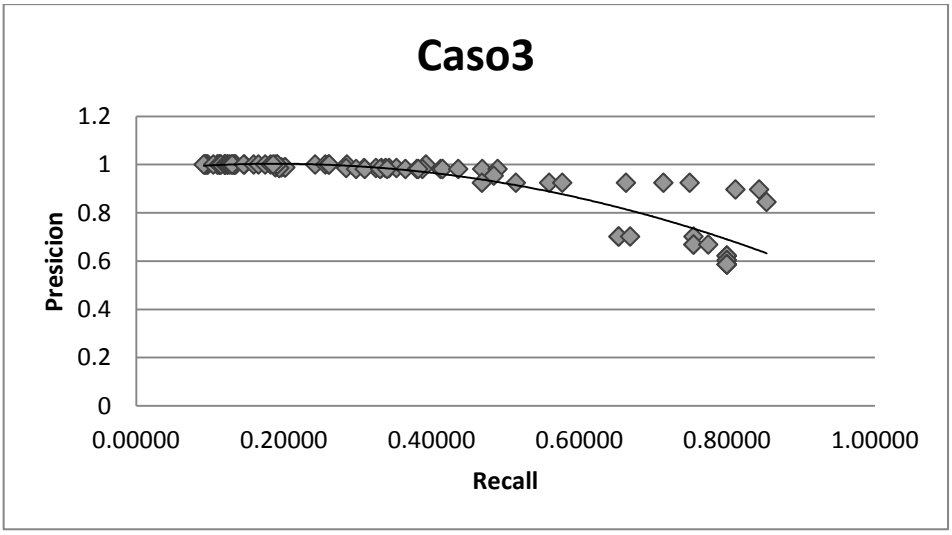
Microaveraging	CASO 1			CASO 2			CASO 3		
Threreshold	**Presicion**	**Recall**	**Overall**	**Presicion**	**Recall**	**Overall**	**Presicion**	**Recall**	**Overall**
0	1	0,43408	0,43408	1	0,04835	0,04835	1	0,39251	0,39251
0,05	1	0,22940	0,22940	1	0,08178	0,08178	1	0,11041	0,11041
0,1	1	0,23658	0,23658	1	0,08268	0,08268	1	0,11332	0,11332
0,15	1	0,24284	0,24284	1	0,08236	0,08236	1	0,11343	0,11343
0,2	1	0,24284	0,24284	1	0,08282	0,08282	1	0,11396	0,11396
0,25	1	0,25568	0,25568	1	0,08788	0,08788	1	0,12024	0,12024
0,3	1	0,25600	0,25600	1	0,08828	0,08828	1	0,12075	0,12075
0,35	1	0,25600	0,25600	1	0,08770	0,08770	1	0,12047	0,12047
0,4	1	0,25827	0,25827	1	0,08651	0,08651	1	0,11975	0,11975
0,45	1	0,26738	0,26738	1	0,08933	0,08933	1	0,12390	0,12390
0,5	1	0,24803	0,24803	1	0,07668	0,07668	1	0,11033	0,11033
0,55	1	0,24980	0,24980	1	0,05964	0,05964	1	0,09730	0,09730
0,6	1	0,24980	0,24980	1	0,05858	0,05858	1	0,09658	0,09658
0,65	1	0,24385	0,24385	1	0,05433	0,05433	1	0,09296	0,09296
0,7	1	0,24452	0,24452	1	0,05476	0,05476	1	0,09394	0,09394

0,75	1	0,24579	0,24579	1	0,05609	0,05609	1	0,09576	0,09576
0,8	1	0,25111	0,25111	1	0,05225	0,05225	1	0,09381	0,09381
0,85	1	0,23578	0,23578	1	0,05140	0,05140	1	0,09175	0,09175
0,9	1	0,26026	0,26026	1	0,06013	0,06013	1	0,10494	0,10494
0,95	1	0,28327	0,28327	1	0,06392	0,06392	1	0,11341	0,11341
1	1	0,28743	0,28743	1	0,06086	0,06086	1	0,11267	0,11267
1,05	1	0,30203	0,30203	1	0,06152	0,06152	1	0,11530	0,11530
1,1	1	0,30663	0,30663	1	0,06486	0,06486	1	0,12090	0,12090
1,15	1	0,31525	0,31525	1	0,06646	0,06646	1	0,12396	0,12396
1,2	1	0,31525	0,31525	1	0,06675	0,06675	1	0,12442	0,12442
1,25	1	0,31735	0,31735	1	0,06835	0,06835	1	0,12698	0,12698
1,3	1	0,31988	0,31988	1	0,06973	0,06973	1	0,12919	0,12919
1,35	1	0,32063	0,32063	1	0,07104	0,07104	1	0,13123	0,13123
1,4	1	0,32597	0,32597	1	0,07172	0,07172	1	0,13246	0,13246
1,45	1	0,32675	0,32675	1	0,07218	0,07218	1	0,13321	0,13321
1,5	1	0,32675	0,32675	1	0,07343	0,07343	1	0,13494	0,13494
1,55	1	0,33848	0,33848	1	0,06806	0,06806	1	0,13375	0,13375
1,6	1	0,33848	0,33848	1	0,06223	0,06223	1	0,13032	0,13032
1,65	1	0,34959	0,34959	NaN	0,07188	NaN	1	0,14695	0,14695
1,7	1	0,35157	0,35157	NaN	0,06635	NaN	1	0,14603	0,14603
1,75	1	0,37385	0,37385	NaN	0,07204	NaN	1	0,15948	0,15948
1,8	1	0,38218	0,38218	NaN	0,07528	NaN	1	0,16581	0,16581
1,85	1	0,40461	0,40461	NaN	0,07780	NaN	1	0,17519	0,17519
1,9	1	0,41704	0,41704	NaN	0,08072	NaN	1	0,18238	0,18238
1,95	1	0,43708	0,43708	NaN	0,08335	NaN	1	0,18851	0,18851
2	0,98333334	0,43839	0,42663	NaN	0,08452	NaN	0,9875	0,18867	0,18341
2,05	0,98333334	0,43839	0,42663	NaN	0,08912	NaN	0,9875	0,19381	0,18826
2,1	0,98333334	0,44585	0,43408	NaN	0,09097	NaN	0,9875	0,19708	0,19153
2,15	0,98333334	0,45610	0,44434	NaN	0,09245	NaN	0,9875	0,20124	0,19568
2,2	0,98333334	0,45610	0,44434	NaN	0,09291	NaN	0,9875	0,20236	0,19680
2,25	0,98181819	0,45645	0,44469	NaN	0,07589	NaN	0,98461538	0,19391	0,18820
2,3	1	0,44815	0,44815	NaN	0,07661	NaN	1	0,19083	0,19083
2,35	1	0,44815	0,44815	NaN	0,07706	NaN	1	0,19140	0,19140
2,4	1	0,43638	0,43638	NaN	0,07729	NaN	1	0,18598	0,18598
2,45	1	0,50474	0,50474	NaN	0,10541	NaN	1	0,24263	0,24263
2,5	1	0,50704	0,50704	NaN	0,10887	NaN	1	0,25641	0,25641
2,55	1	0,49833	0,49833	NaN	0,11647	NaN	1	0,26074	0,26074
2,6	1	0,50474	0,50474	NaN	0,09786	NaN	1	0,26170	0,26170
2,65	1	0,52346	0,52346	NaN	0,11043	NaN	1	0,28544	0,28544
2,7	0,98	0,53466	0,52514	NaN	0,09089	NaN	0,98461538	0,28483	0,28120
2,75	0,98	0,60126	0,59073	NaN	0,09883	NaN	0,98461538	0,30879	0,30495
2,8	0,98	0,61241	0,60064	NaN	0,10396	NaN	0,98461538	0,32514	0,32097
2,85	0,98	0,62652	0,61319	NaN	0,10507	NaN	0,98461538	0,33235	0,32801
2,9	0,98	0,62652	0,61319	NaN	0,10814	NaN	0,98461538	0,33793	0,33359
2,95	0,98	0,62652	0,61319	NaN	0,10947	NaN	0,98461538	0,34252	0,33798
3	0,98	0,64015	0,62681	NaN	0,10947	NaN	0,98461538	0,35256	0,34802

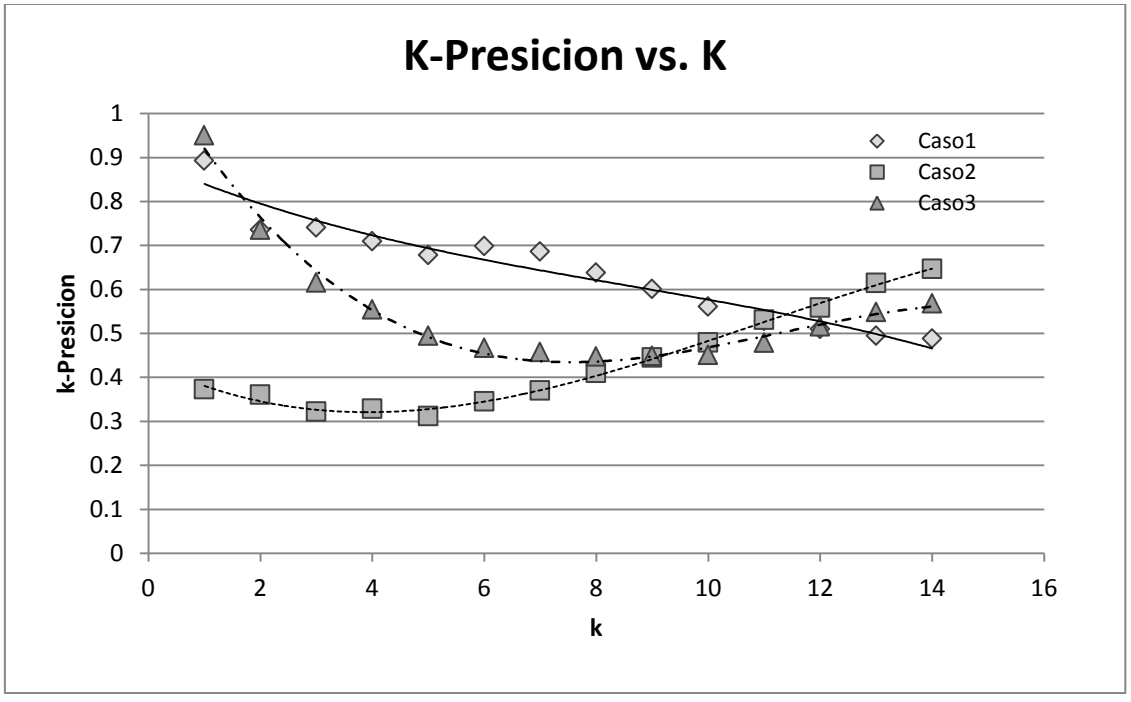
3,05	0,97777778	0,58737	0,57309	NaN	0,07513	NaN	0,98181819	0,29821	0,29345
3,1	0,97777778	0,61271	0,59843	NaN	0,06204	NaN	0,98181819	0,30966	0,30478
3,15	0,97777778	0,64100	0,62671	NaN	0,05470	NaN	0,98181819	0,33060	0,32560
3,2	0,97777778	0,64100	0,62671	NaN	0,05641	NaN	0,98181819	0,33996	0,33470
3,25	0,97777778	0,66044	0,64615	NaN	0,06476	NaN	0,98181819	0,36472	0,35900
3,3	0,97777778	0,66044	0,64615	NaN	0,07117	NaN	0,98181819	0,38801	0,38176
3,35	0,97777778	0,62967	0,61538	NaN	0,07174	NaN	0,98181819	0,38028	0,37403
3,4	0,97777778	0,62967	0,61538	NaN	0,06977	NaN	0,98181819	0,38209	0,37564
3,45	0,97777778	0,62967	0,61538	NaN	0,08783	NaN	0,98181819	0,41247	0,40532
3,5	0,97777778	0,62967	0,61538	NaN	0,08818	NaN	0,98181819	0,41526	0,40785
3,55	0,97777778	0,64872	0,63205	NaN	0,09848	NaN	0,98181819	0,43655	0,42822
3,6	0,97777778	0,64872	0,63205	NaN	0,10260	NaN	0,98181819	0,46906	0,46036
3,65	0,97777778	0,67727	0,66061	NaN	0,11619	NaN	0,98181819	0,48980	0,48071
3,7	0,94920635	0,67370	0,62846	NaN	0,11619	NaN	0,95324676	0,48424	0,45015
3,75	0,92063493	0,68398	0,62056	NaN	0,07778	NaN	0,92467534	0,46842	0,41961
3,8	0,92063493	0,69610	0,63117	NaN	NaN	NaN	0,92467534	0,51451	0,46165
3,85	0,92063493	0,76303	0,69152	NaN	NaN	NaN	0,92467534	0,55925	0,50111
3,9	0,92063493	0,76303	0,69152	NaN	NaN	NaN	0,92467534	0,57712	0,51659
3,95	0,92063493	0,83222	0,75389	NaN	NaN	NaN	0,92467534	0,66364	0,59475
4	0,92063493	0,86556	0,78722	NaN	NaN	NaN	0,92467534	0,71429	0,64444
4,05	0,92063493	0,86556	0,78722	NaN	NaN	NaN	0,92467534	0,74987	0,68003
4,1	0,8920635	0,91000	0,78500	NaN	NaN	NaN	0,89610391	0,81154	0,69115
4,15	0,8920635	0,92778	0,80056	NaN	NaN	NaN	0,89610391	0,84394	0,72227
4,2	0,82984128	0,92500	0,72000	NaN	NaN	NaN	0,84458874	0,85364	0,67227
4,25	0,68698413	0,72500	0,60000	NaN	NaN	NaN	0,70173159	0,65364	0,55227
4,3	0,68698413	0,72500	0,60000	NaN	NaN	NaN	0,70173159	0,65364	0,55227
4,35	0,68698413	0,72500	0,60000	NaN	NaN	NaN	0,70173159	0,66919	0,56783
4,4	0,68698413	0,77500	0,64167	NaN	NaN	NaN	0,70173159	0,75500	0,64167
4,45	0,64698413	0,77500	0,55833	NaN	NaN	NaN	0,66839827	0,75500	0,58167
4,5	0,64698413	0,77500	0,55833	NaN	NaN	NaN	0,66839827	0,75500	0,58167
4,55	0,64698413	0,80000	0,57619	NaN	NaN	NaN	0,66839827	0,77500	0,59722
4,6	0,59619049	0,80000	0,48667	NaN	NaN	NaN	0,62164504	0,80000	0,54500
4,65	0,59619049	0,80000	0,48667	NaN	NaN	NaN	0,62164504	0,80000	0,54500
4,7	0,59619049	0,80000	0,48667	NaN	NaN	NaN	0,62164504	0,80000	0,54500
4,75	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,60346321	0,80000	0,50571
4,8	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,60346321	0,80000	0,50571
4,85	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,5852814	0,80000	0,45333
4,9	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,5852814	0,80000	0,45333
4,95	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,5852814	0,80000	0,45333
5	0,57396827	0,80000	0,42667	NaN	NaN	NaN	0,5852814	0,80000	0,45333


Caso1
Presicion
1.2
1
0.8
0.6
0.4
0.2
0
0.00000
0.20000
0.40000
0.60000
0.80000
1.00000
Recall

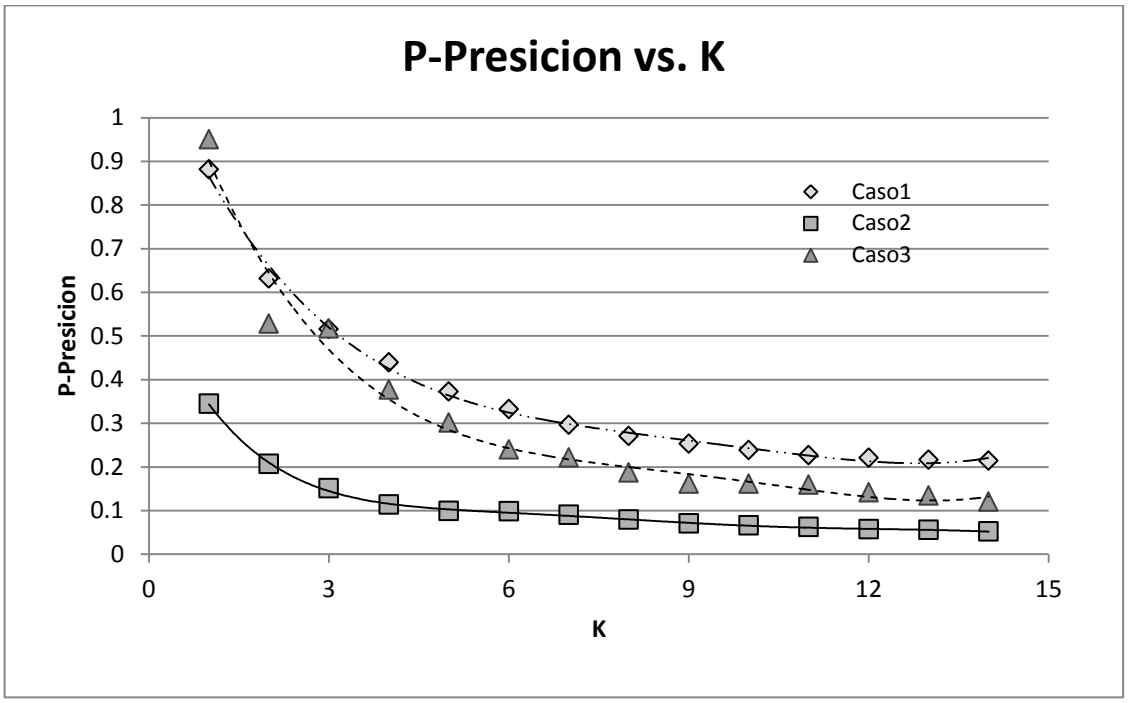

Caso2
Presicion
1.2
1
0.8
0.6
0.4
0.2
0
0.00000
0.02000
0.04000
0.06000
0.08000
0.10000
Recall


Caso3
Presicion
1.2
1
0.8
0.6
0.4
0.2
0
0.00000
0.20000
0.40000
0.60000
0.80000
1.00000
Recall

K	K-Presicion vs. K		
	Caso1	Caso2	Caso3
1	0,89285715	0,37285715	0,95
2	0,73571429	0,36071429	0,73571429
3	0,7404762	0,32190477	0,61523812
4	0,70964285	0,32857143	0,55428571
5	0,67800001	0,312	0,49428572
6	0,69809522	0,345	0,46714286
7	0,68571429	0,37040817	0,45734695
8	0,6382653	0,41017857	0,4475
9	0,60064627	0,44539684	0,44904763
10	0,56083674	0,47928572	0,45085714
11	0,52826529	0,53103895	0,47857144
12	0,50991033	0,55892856	0,51571428
13	0,49437919	0,61472531	0,54879123
14	0,48809976	0,64693878	0,56785714

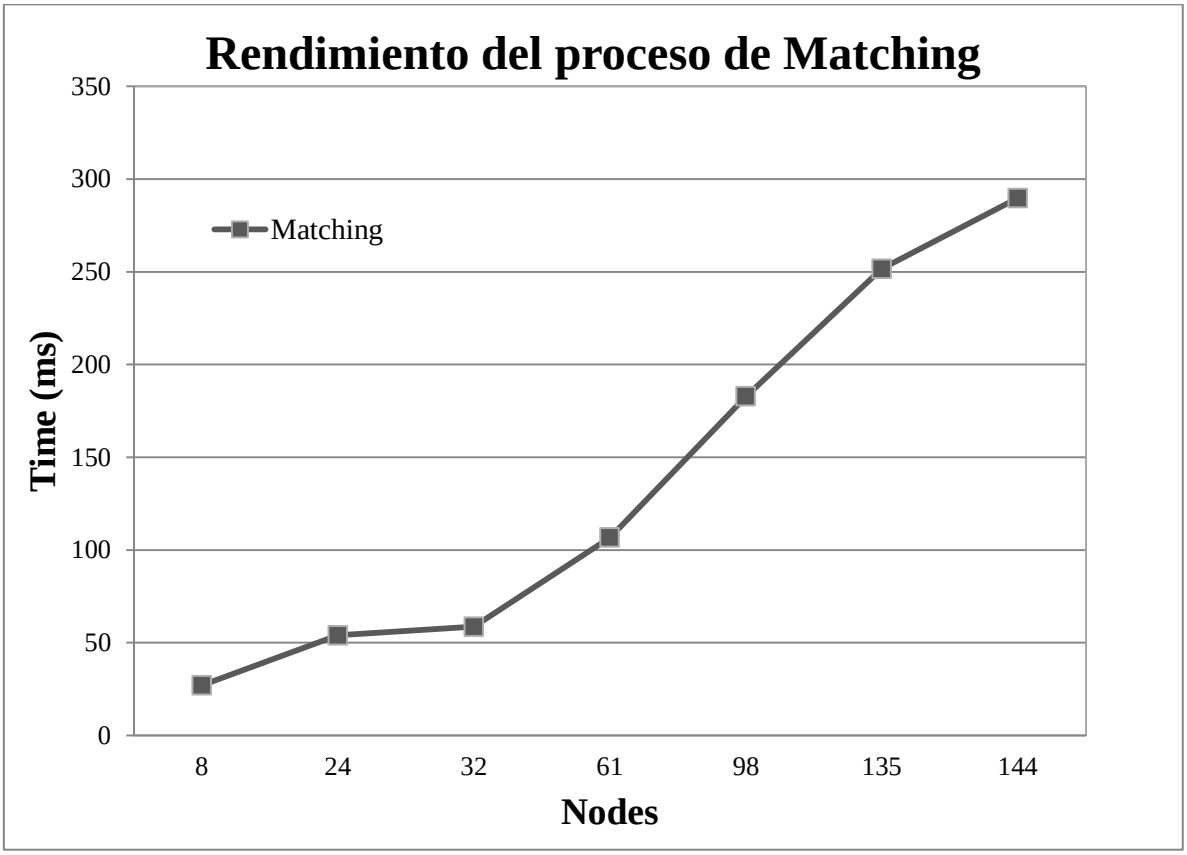


K	P-Presicion vs. K		
	Caso1	Caso2	Caso3
1	0,881428574	0,3442857	0,95
2	0,631428572	0,2064286	0,5278571
3	0,515714296	0,1509524	0,5171429
4	0,438928566	0,1132143	0,3764286
5	0,372571432	0,0985714	0,3011429
6	0,332142864	0,0983333	0,2397619
7	0,296122452	0,09	0,2212245
8	0,270739794	0,07875	0,1869643
9	0,25280329	0,07	0,1611111
10	0,238454082	0,0658571	0,1608571
11	0,226713822	0,0624675	0,1587013
12	0,220869666	0,0572619	0,1411905
13	0,215924612	0,0559341	0,1342857
14	0,213883794	0,0519388	0,1204082



D.2.2 Prueba de Rendimiento ER1

Nodes	Matching time (ms)					Total
	Dominio1	Dominio2	Dominio3	Dominio4	Dominio5	
8	10	10,2	35,6	40,4	39,2	**27,08**
24	20,8	12,8	71,4	89,6	75,6	**54,04**
32	22,4	13,6	78	104,6	74,8	**58,68**
61	49	46,6	160,2	152,8	125	**106,72**
98	99,2	91,4	236	256,2	231,6	**182,88**
135	158,8	128,4	353,4	332,6	285	**251,64**
144	184,4	161	417,4	373,8	312,4	**289,8**



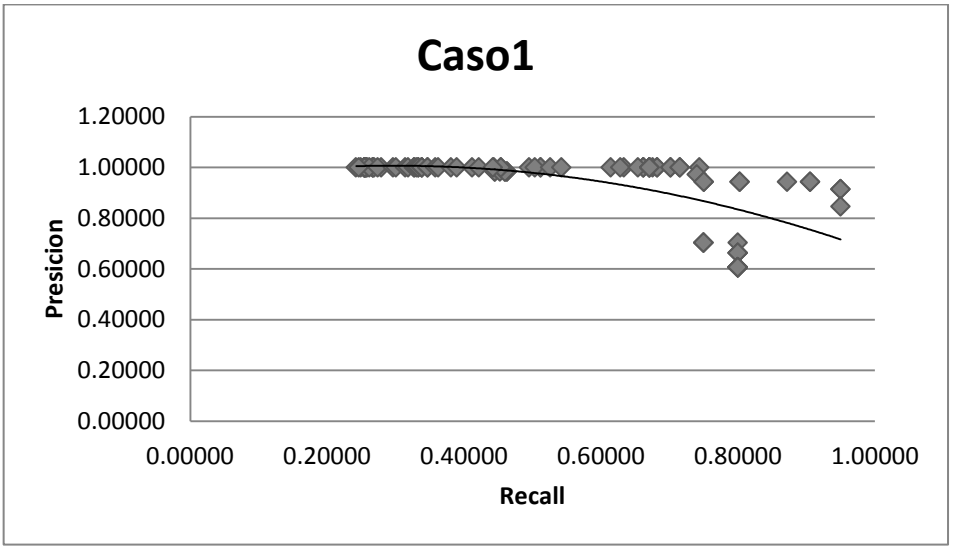
D.3 Recuperación de Actividades Básicas BPEL adaptado al Contexto (SeMatch-Context)

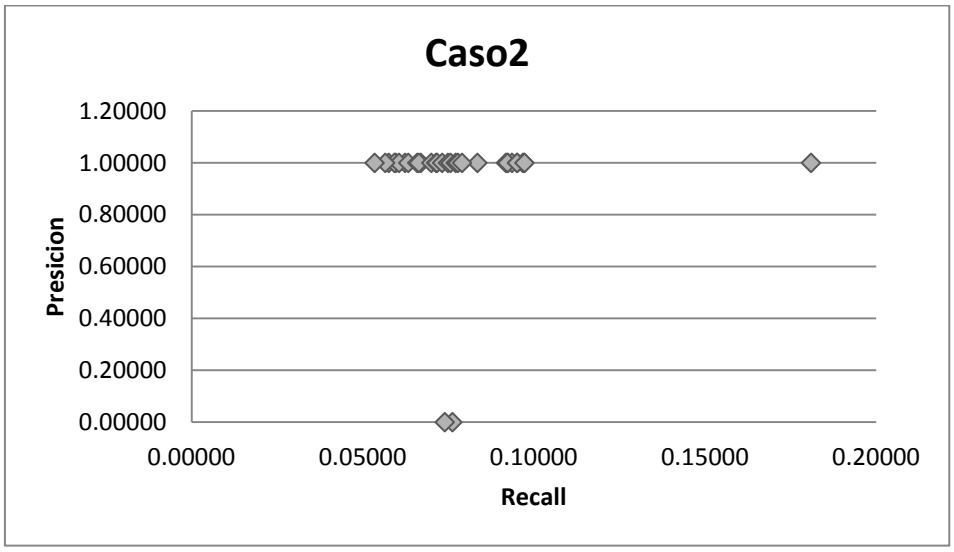
D.3.1 Prueba de Calidad de los Resultados CC1

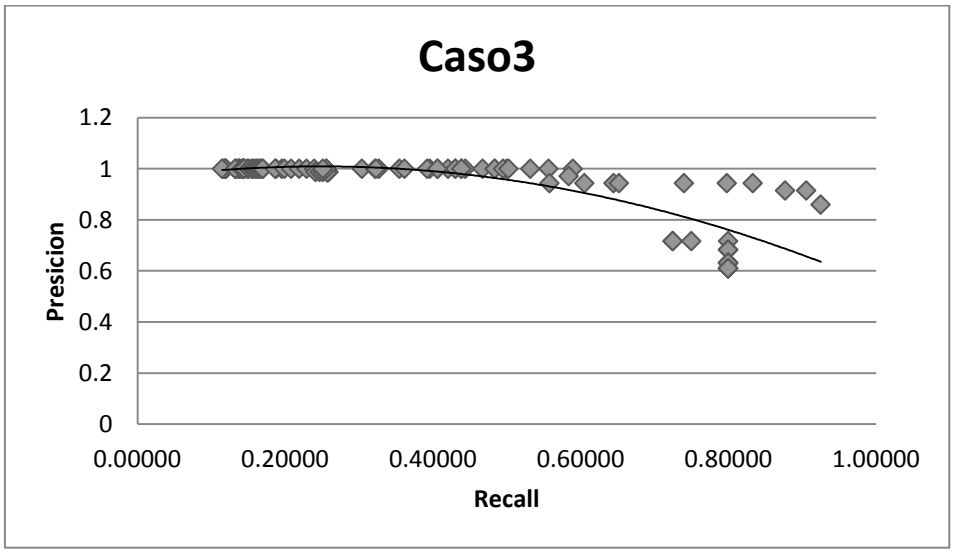
Microaveraging	CASO 1			CASO 2			CASO 3		
Threreshold	**Presicion**	**Recall**	**Overall**	**Presicion**	**Recall**	**Overall**	**Presicion**	**Recall**	**Overall**
0	1,00000	0,25500	0,25500	1,00000	0,18101	0,18101	1	0,19582	0,19582
0,05	1,00000	0,24217	0,24217	1,00000	0,09248	0,09248	1	0,13487	0,13487
0,1	1,00000	0,24935	0,24935	1,00000	0,09367	0,09367	1	0,13833	0,13833
0,15	1,00000	0,25561	0,25561	1,00000	0,09193	0,09193	1	0,13770	0,13770
0,2	1,00000	0,25561	0,25561	1,00000	0,09239	0,09239	1	0,13821	0,13821
0,25	1,00000	0,26671	0,26671	1,00000	0,09690	0,09690	1	0,14415	0,14415
0,3	1,00000	0,26671	0,26671	1,00000	0,09727	0,09727	1	0,14457	0,14457
0,35	1,00000	0,26671	0,26671	1,00000	0,09522	0,09522	1	0,14350	0,14350
0,4	1,00000	0,26898	0,26898	1,00000	0,09522	0,09522	1	0,14399	0,14399
0,45	1,00000	0,27915	0,27915	1,00000	0,09736	0,09736	1	0,14889	0,14889
0,5	1,00000	0,25561	0,25561	1,00000	0,08363	0,08363	1	0,13283	0,13283
0,55	1,00000	0,25827	0,25827	1,00000	0,06241	0,06241	1	0,11910	0,11910
0,6	1,00000	0,25827	0,25827	1,00000	0,05973	0,05973	1	0,11763	0,11763
0,65	1,00000	0,25410	0,25410	1,00000	0,05773	0,05773	1	0,11634	0,11634
0,7	1,00000	0,25410	0,25410	1,00000	0,05942	0,05942	1	0,11859	0,11859
0,75	1,00000	0,25537	0,25537	1,00000	0,06065	0,06065	1	0,12043	0,12043
0,8	1,00000	0,26192	0,26192	1,00000	0,05668	0,05668	1	0,11906	0,11906
0,85	1,00000	0,24622	0,24622	1,00000	0,05357	0,05357	1	0,11466	0,11466
0,9	1,00000	0,27405	0,27405	1,00000	0,06337	0,06337	1	0,13256	0,13256
0,95	1,00000	0,29646	0,29646	1,00000	0,06708	0,06708	1	0,14120	0,14120
1	1,00000	0,30062	0,30062	1,00000	0,06605	0,06605	1	0,14251	0,14251
1,05	1,00000	0,31394	0,31394	1,00000	0,06651	0,06651	1	0,14474	0,14474
1,1	1,00000	0,31854	0,31854	1,00000	0,07009	0,07009	1	0,15122	0,15122
1,15	1,00000	0,32716	0,32716	1,00000	0,07156	0,07156	1	0,15544	0,15544
1,2	1,00000	0,32716	0,32716	1,00000	0,07185	0,07185	1	0,15590	0,15590
1,25	1,00000	0,33074	0,33074	1,00000	0,07333	0,07333	1	0,15843	0,15843
1,3	1,00000	0,33327	0,33327	1,00000	0,07499	0,07499	1	0,16126	0,16126
1,35	1,00000	0,33327	0,33327	1,00000	0,07573	0,07573	1	0,16235	0,16235
1,4	1,00000	0,33860	0,33860	1,00000	0,07727	0,07727	1	0,16530	0,16530
1,45	1,00000	0,33860	0,33860	1,00000	0,07786	0,07786	1	0,16616	0,16616
1,5	1,00000	0,33860	0,33860	1,00000	0,07908	0,07908	1	0,16771	0,16771
1,55	1,00000	0,34685	0,34685	#¡VALOR!	0,07634	#¡VALOR!	1	0,17040	0,17040
1,6	1,00000	0,34685	0,34685	#¡VALOR!	0,07409	#¡VALOR!	1	0,16961	0,16961
1,65	1,00000	0,35796	0,35796	#¡VALOR!	0,08489	#¡VALOR!	1	0,18804	0,18804
1,7	1,00000	0,36211	0,36211	#¡VALOR!	0,07881	#¡VALOR!	1	0,18660	0,18660
1,75	1,00000	0,38107	0,38107	#¡VALOR!	0,08388	#¡VALOR!	1	0,19912	0,19912

1,8	1,00000	0,38940	0,38940	#¡VALOR!	0,08995	#¡VALOR!	1	0,20822	0,20822
1,85	1,00000	0,41183	0,41183	#¡VALOR!	0,09286	#¡VALOR!	1	0,21901	0,21901
1,9	1,00000	0,42176	0,42176	#¡VALOR!	0,09873	#¡VALOR!	1	0,22937	0,22937
1,95	1,00000	0,44389	0,44389	#¡VALOR!	0,10211	#¡VALOR!	1	0,23940	0,23940
2	0,98333	0,44521	0,43344	#¡VALOR!	0,10373	#¡VALOR!	0,9875	0,24124	0,23597
2,05	0,98333	0,44521	0,43344	#¡VALOR!	0,10884	#¡VALOR!	0,9875	0,24688	0,24132
2,1	0,98333	0,45266	0,44090	#¡VALOR!	0,11168	#¡VALOR!	0,9875	0,25144	0,24589
2,15	0,98333	0,46292	0,45115	#¡VALOR!	0,11411	#¡VALOR!	0,9875	0,25807	0,25252
2,2	0,98333	0,46292	0,45115	#¡VALOR!	0,11411	#¡VALOR!	0,9875	0,25929	0,25374
2,25	0,98182	0,46018	0,44841	#¡VALOR!	0,10079	#¡VALOR!	0,98461538	0,25756	0,25184
2,3	1,00000	0,45421	0,45421	#¡VALOR!	0,10165	#¡VALOR!	1	0,25588	0,25588
2,35	1,00000	0,45421	0,45421	#¡VALOR!	0,10165	#¡VALOR!	1	0,25588	0,25588
2,4	1,00000	0,44244	0,44244	#¡VALOR!	0,10244	#¡VALOR!	1	0,25111	0,25111
2,45	1,00000	0,51175	0,51175	#¡VALOR!	0,13004	#¡VALOR!	1	0,30411	0,30411
2,5	1,00000	0,51175	0,51175	#¡VALOR!	0,13989	#¡VALOR!	1	0,32192	0,32192
2,55	1,00000	0,49509	0,49509	#¡VALOR!	0,14954	#¡VALOR!	1	0,32722	0,32722
2,6	1,00000	0,50376	0,50376	#¡VALOR!	0,11939	#¡VALOR!	1	0,32317	0,32317
2,65	1,00000	0,52588	0,52588	#¡VALOR!	0,13553	#¡VALOR!	1	0,35466	0,35466
2,7	1,00000	0,54228	0,54228	#¡VALOR!	0,12039	#¡VALOR!	1	0,36190	0,36190
2,75	1,00000	0,61422	0,61422	#¡VALOR!	0,13039	#¡VALOR!	1	0,39606	0,39606
2,8	1,00000	0,63380	0,63380	#¡VALOR!	0,13039	#¡VALOR!	1	0,40690	0,40690
2,85	1,00000	0,66208	0,66208	#¡VALOR!	0,13333	#¡VALOR!	1	0,42072	0,42072
2,9	1,00000	0,66208	0,66208	#¡VALOR!	0,14000	#¡VALOR!	1	0,43085	0,43085
2,95	1,00000	0,66208	0,66208	#¡VALOR!	0,14000	#¡VALOR!	1	0,43085	0,43085
3	1,00000	0,67570	0,67570	#¡VALOR!	0,14000	#¡VALOR!	1	0,44433	0,44433
3,05	1,00000	0,62864	0,62864	#¡VALOR!	0,10589	#¡VALOR!	1	0,39244	0,39244
3,1	1,00000	0,65398	0,65398	#¡VALOR!	0,08142	#¡VALOR!	1	0,40643	0,40643
3,15	1,00000	0,68226	0,68226	#¡VALOR!	0,08636	#¡VALOR!	1	0,43902	0,43902
3,2	1,00000	0,68226	0,68226	#¡VALOR!	0,08636	#¡VALOR!	1	0,43902	0,43902
3,25	1,00000	0,70171	0,70171	#¡VALOR!	0,10111	#¡VALOR!	1	0,46740	0,46740
3,3	1,00000	0,70171	0,70171	#¡VALOR!	0,11977	#¡VALOR!	1	0,50054	0,50054
3,35	1,00000	0,67094	0,67094	#¡VALOR!	0,11977	#¡VALOR!	1	0,48387	0,48387
3,4	1,00000	0,67094	0,67094	#¡VALOR!	0,11977	#¡VALOR!	1	0,48387	0,48387
3,45	1,00000	0,67094	0,67094	#¡VALOR!	0,12937	#¡VALOR!	1	0,49508	0,49508
3,5	1,00000	0,67094	0,67094	#¡VALOR!	0,13889	#¡VALOR!	1	0,50258	0,50258
3,55	1,00000	0,71538	0,71538	#¡VALOR!	0,14603	#¡VALOR!	1	0,53202	0,53202
3,6	1,00000	0,71538	0,71538	#¡VALOR!	0,14881	#¡VALOR!	1	0,55687	0,55687
3,65	1,00000	0,74394	0,74394	#¡VALOR!	0,17167	#¡VALOR!	1	0,58977	0,58977
3,7	0,97143	0,74037	0,71180	#¡VALOR!	0,17167	#¡VALOR!	0,97142857	0,58421	0,55921
3,75	0,94286	0,75065	0,70390	#¡VALOR!	0,11333	#¡VALOR!	0,94285715	0,55830	0,51791
3,8	0,94286	0,75065	0,70390	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,60518	0,56122
3,85	0,94286	0,80303	0,75152	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,64466	0,59594
3,9	0,94286	0,80303	0,75152	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,65235	0,60235
3,95	0,94286	0,87222	0,81389	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,74000	0,68444
4	0,94286	0,90556	0,84722	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,79810	0,74254
4,05	0,94286	0,90556	0,84722	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,94285715	0,83333	0,77778

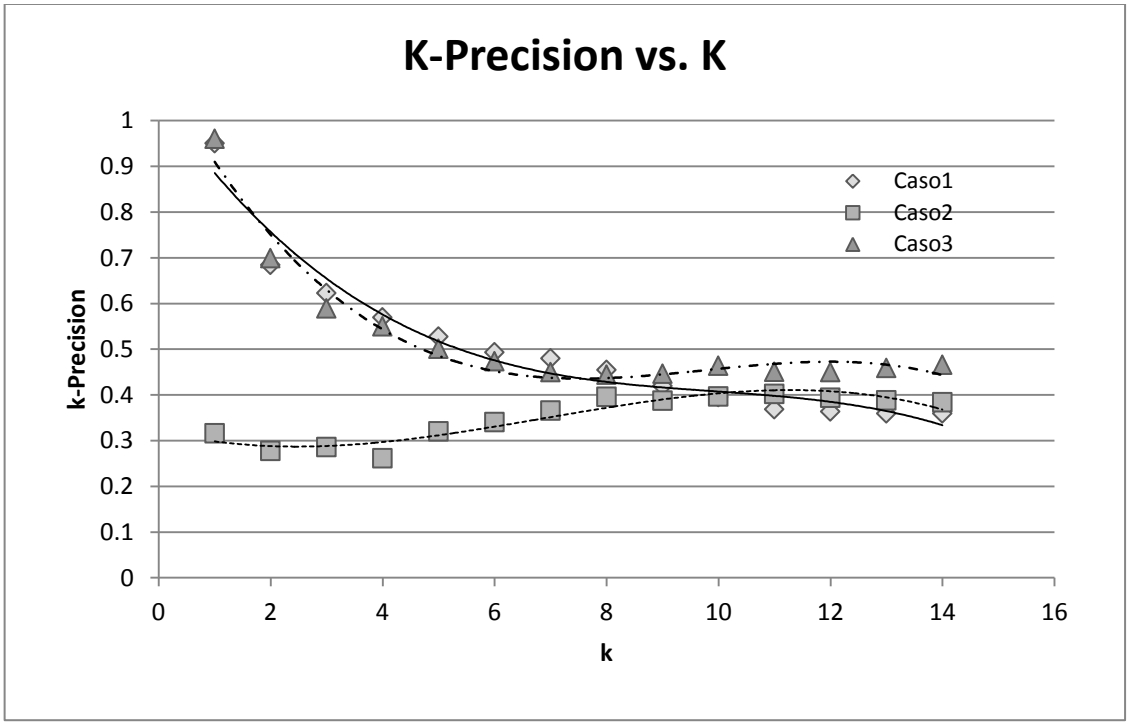
4,1	0,91429	0,95000	0,84500	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,91428572	0,87727	0,77227
4,15	0,91429	0,95000	0,84500	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,91428572	0,90556	0,80056
4,2	0,84571	0,95000	0,76167	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,85873016	0,92500	0,75500
4,25	0,70286	0,75000	0,64167	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,71587301	0,72500	0,63500
4,3	0,70286	0,75000	0,64167	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,71587301	0,72500	0,63500
4,35	0,70286	0,75000	0,64167	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,71587301	0,75000	0,66000
4,4	0,70286	0,80000	0,68333	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,71587301	0,80000	0,70167
4,45	0,66286	0,80000	0,60000	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,68253969	0,80000	0,64167
4,5	0,66286	0,80000	0,60000	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,68253969	0,80000	0,64167
4,55	0,66286	0,80000	0,60000	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,68253969	0,80000	0,64167
4,6	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,63174605	0,80000	0,56286
4,65	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,63174605	0,80000	0,56286
4,7	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,63174605	0,80000	0,56286
4,75	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,63174605	0,80000	0,56286
4,8	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,63174605	0,80000	0,56286
4,85	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,60952383	0,80000	0,52000
4,9	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,60952383	0,80000	0,52000
4,95	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,60952383	0,80000	0,52000
5	0,60571	0,80000	0,50667	#¡VALOR!	#¡VALOR!	#¡VALOR!	0,60952383	0,80000	0,52000



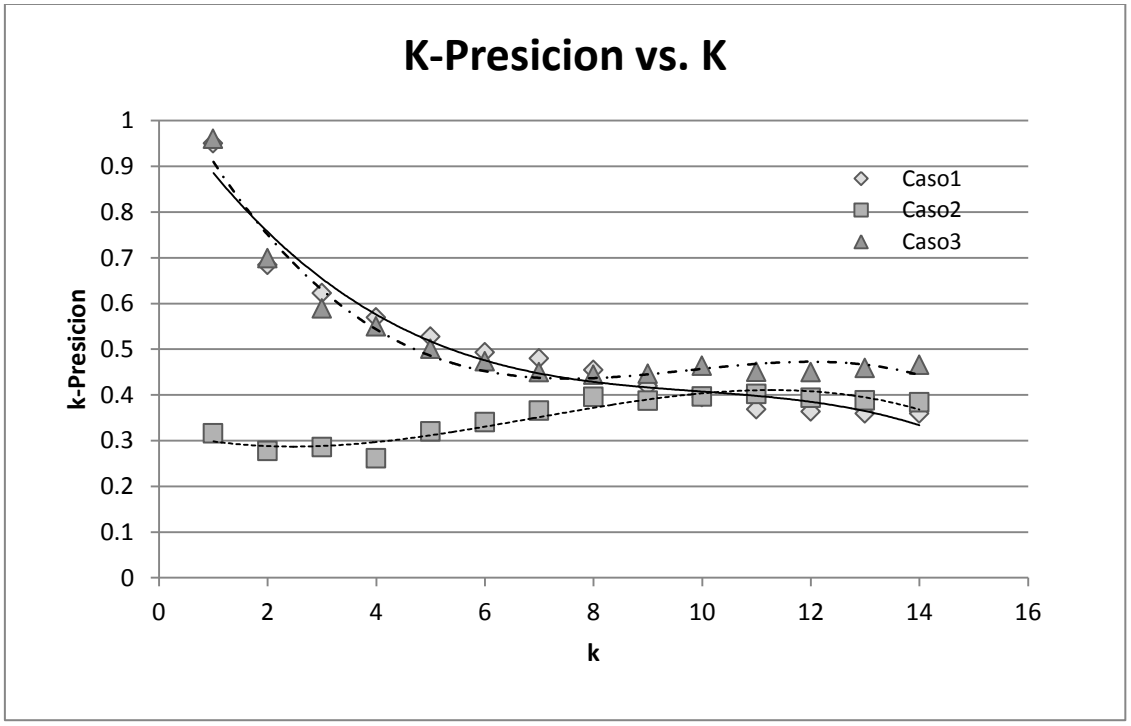

Caso2
1.20000
1.00000
0.80000
0.60000
0.40000
0.20000
0.00000
Presicion
0.00000
0.05000
0.10000
0.15000
0.20000
Recall


Caso3
1.2
1
0.8
0.6
0.4
0.2
0
Presicion
0.00000
0.20000
0.40000
0.60000
0.80000
1.00000
Recall

K	K-Precision		
	Caso1	Caso2	Caso3
1	0,95	0,31571429	0,96
2	0,68428572	0,27714286	0,69857143
3	0,62238096	0,28571429	0,58857145
4	0,56928571	0,26142857	0,54964285
5	0,52742857	0,32	0,50085715
6	0,49285714	0,34047619	0,47333333
7	0,47959185	0,36571429	0,44959184
8	0,45423469	0,39535714	0,44392857
9	0,4192347	0,38746032	0,44619048
10	0,39523469	0,39628571	0,46342857
11	0,3683256	0,40194805	0,45025974
12	0,36347713	0,39345237	0,44916666
13	0,35937456	0,38780221	0,45835165
14	0,35937456	0,38367348	0,46561224

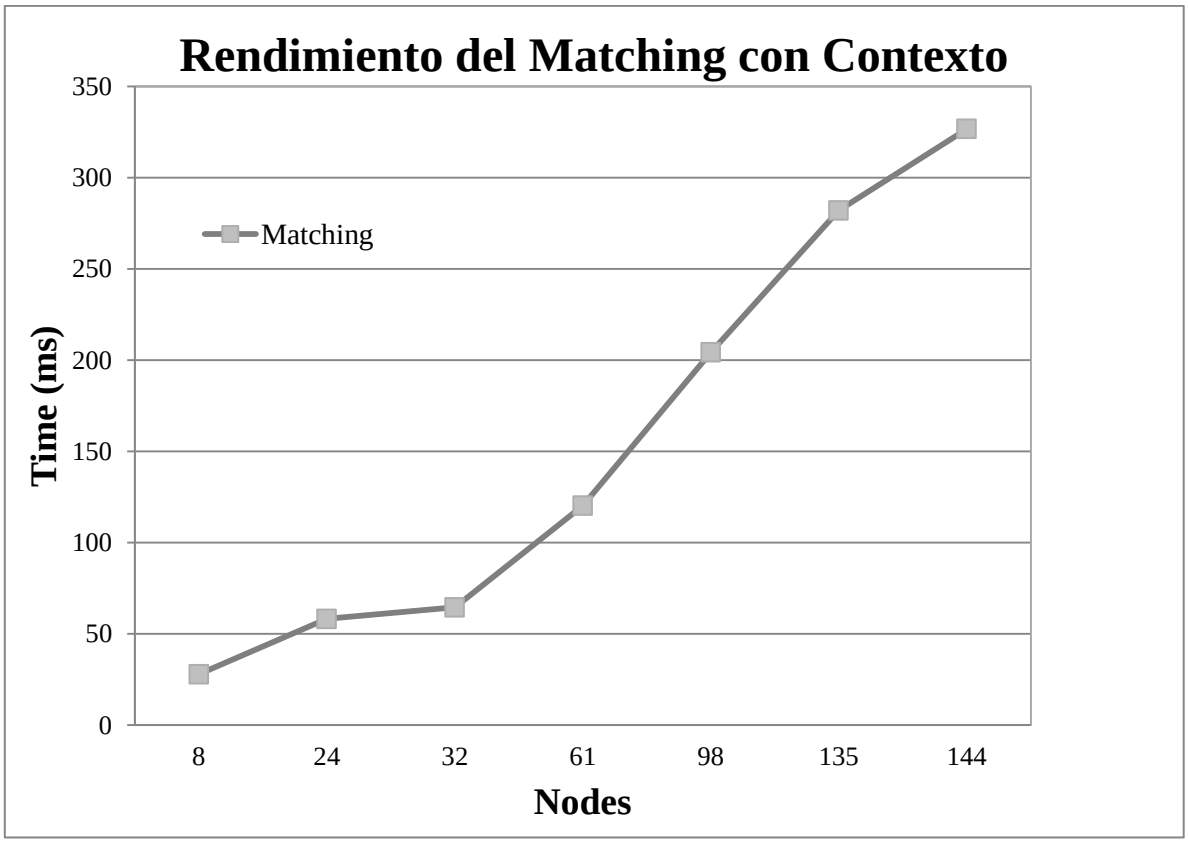


K	P-Precision		
	Caso1	Caso2	Caso3
1	0,91	0,3157143	0,96
2	0,59428572	0,1778571	0,5535714
3	0,466190486	0,1414286	0,462381
4	0,38214286	0,1132143	0,3464286
5	0,313714286	0,0985714	0,2634286
6	0,27476191	0,0971429	0,2711905
7	0,241224494	0,0832653	0,242449
8	0,22015306	0,0728571	0,1871429
9	0,203486396	0,0647619	0,1819048
10	0,19015306	0,0622857	0,1757143
11	0,17924397	0,0566234	0,1668831
12	0,174092454	0,0519048	0,1488095
13	0,16973348	0,050989	0,1327473
14	0,16973348	0,0473469	0,1182653



D.3.1 Prueba de rendimiento CR1:

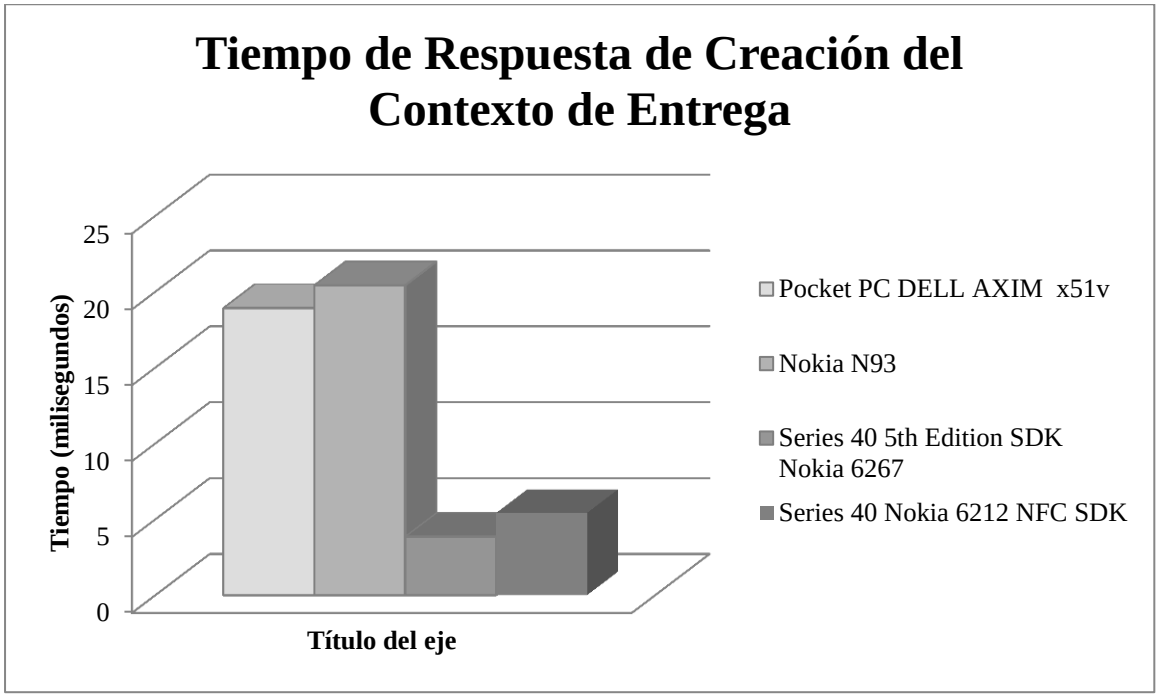
Nodes	Matching time (ms)					Total
	Dominio1	Dominio1	Dominio1	Dominio1	Dominio1	
8	6,4	11,6	39	42,6	39,8	**27,88**
24	22	12,2	76,4	97,6	83	**58,24**
32	22,4	12,6	87	112,6	87,8	**64,48**
61	50,2	50,2	191,2	167	142,4	**120,2**
98	108,8	103,2	259	290,8	259,6	**204,28**
135	156,6	145,2	400,4	382,2	325,8	**282,04**
144	192,6	180,8	492	417,2	351	**326,72**



D.4 Obtención del Contexto del Dispositivo Móvil

D.4.1 Prueba de Rendimiento OR1:

Dispositivo	Prueba1	Prueba1	Prueba1	Prueba1	Prueba1	Prueba1	Total
Pocket PC DELL AXIM x51v	19	18	17,5	20,8	19,3	19	18,92
Nokia N93	21	20,3	20,8	21,2	20	19,8	20,42
Series 40 5th Edition SDK Nokia 6267	4	3,8	3,5	3,9	4,1	4	3,86
Series 40 Nokia 6212 NFC SDK	5	5,2	5,1	4,8	5,1	6,9	5,42



D.5 Recuperación de Servicios A través de Terminales Móviles

D.5.1 Prueba de Rendimiento RR1

Nodes	Total Time (millisecond)			
	Series 40 Nokia 6212 NFC SDK	Pocket PC DELL AXIM x51v	Nokia N93	Series 40 5th Edition Nokia 6267 SDK
8	27,88	127,88	137,88	30
24	58,24	158,24	148,24	62
32	64,48	184,48	204,48	72
61	120,2	245,2	255,2	124
98	204,28	334,28	314,28	210
135	282,04	412,04	422,04	290
144	326,72	466,72	476,72	335