DESCUBRIMIENTO AUTOMÁTICO DE PROCESOS DE NEGOCIO BASADO EN SEMÁNTICA DE COMPORTAMIENTO

CRISTHIAN NICOLÁS FIGUEROA MARTÍNEZ

Tesis de Maestría en Ingeniería Telemática

Director:
Juan Carlos Corrales
Doctor en Ciencias de la Computación

Universidad del Cauca
Facultad de Ingeniería en Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación en Aplicaciones y Servicios sobre Internet
Popayán, noviembre de 2011

CRISTHIAN NICOLÁS FIGUEROA MARTÍNEZ

DESCUBRIMIENTO AUTOMÁTICO DE PROCESOS DE NEGOCIO BASADO EN SEMÁNTICA DEL COMPORTAMIENTO

Tesis presentada a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del Título de

Magíster en:
Ingeniería Telemática

Director:
PhD. Juan Carlos Corrales

Popayán
2011

A Dios, el amor puro y eterno
A mis padres, la inspiración de mi pensamiento
A mis hermanos Camilo, Carolina y Dayana, la confianza plena y sincera
A mis amigos, el apoyo incondicional

Agradecimientos

Quiero expresar mis más sinceros agradecimientos al Dr. Juan Carlos Corrales, por su excelente dirección, por sus aportes tan importantes y por su apoyo incondicional durante toda mi formación profesional. A todos los integrantes del IPET, a Lorena Vidal y Patricia Campo por su valiosa colaboración y a los ingenieros Dr. Álvaro Rendón Gallón, Mg. Eduardo Rojas, Armando Ordóñez, Francisco Martínez, Javier Hurtado y Oscar Caicedo por sus conocimientos y consejos. A los estudiantes que trabajaron como soporte de este trabajo de grado, los ingenieros Adriana Bastidas, Laura Sandino, David Corchuelo, Leandro Ordóñez y Daniel Rivas.

Quiero, además extender mis agradecimientos a COLCIENCIAS por el apoyo financiero a través del programa Joven Investigador; a COOPEN por la beca de pasantía investigativa en la Universidad Politécnica de Turín; y a todos los miembros del grupo de investigación SoftEng en especial al Dr. Maurizio Morisio, el Mg. Oscar Rodríguez y el Dr. Felipe Mejía por brindarme su amistad y soporte durante mi estadía en el Politécnico de Turín.

Finalmente quiero agradecer a todos mis amigos integrantes del Grupo de investigación en Ingeniería Telemática, por mantener en mí encendido el espíritu investigativo y el valor para continuar con mis estudios.

Resumen Estructurado

Antecedentes: La búsqueda de procesos de negocio (BP) se ha convertido en un campo crítico en muchas áreas, debido a la dificultad de encontrar servicios que se adapten exactamente a las exigencias de los usuarios. Por esta razón se puede llegar a incurrir en un consumo innecesario de recursos destinados a la búsqueda del BP adecuado. De esta manera, el desarrollo de nuevos métodos de búsqueda, intuitivos, dotados de inteligencia artificial, basados en semántica y que reconozcan lo que realmente el usuario necesita encontrar, permite agilizar el despliegue y configuración de nuevos BP y se convierten en un área importante de la I+D.

Objetivos: Definir mecanismos para el descubrimiento de procesos de negocio considerando requerimientos de búsqueda que tengan en cuenta la semántica del comportamiento**.**

Métodos: Se propone un entorno denominado BeMantics para la recuperación de BP, el cual está compuesto por dos módulos principales: un repositorio con mecanismos de pre-correspondencia basada en semántica del comportamiento y un mecanismo de correspondencia estructural y semántica el cual refina los resultados del repositorio utilizando un algoritmo de corrección de errores. Este algoritmo ejecuta operaciones de edición sobre un BP con el fin de hacerlo tan similar como sea posible a otro BP de consulta. Para analizar los resultados del entorno BeMantics se generó una plataforma Web de evaluación de pertinencia la cual permitió que un grupo de jueces humanos emita juicios de relevancia sobre un conjunto de 100 BP. Estos juicios de relevancia permitieron obtener un sub-conjunto de los BP considerados como relevantes, es decir, aquellos BP que se esperaría que una herramienta automática recupere de acuerdo a un BP de consulta.

Resultados: La presente propuesta entregó como resultados, la creación de un modelo de evaluación de pertinencia de la recuperación de BP; un repositorio de BP basado en semántica del comportamiento; un analizador estructural y semántico de correspondencia entre BP; un conjunto de 100BP de los dominios de geo-procesamiento y telecomunicaciones; y una ontología de patrones de control de flujo.

Conclusiones: El entorno de descubrimiento de BP propuesto en el presente trabajo demostró que el uso de la semántica en la comparación de tareas de procesos de negocio permite mejorar levemente la precisión de la correspondencia, además el uso de ontologías de dominios muy específicos limita la capacidad semántica que se podría obtener al utilizar ontologías de dominios más generales; no obstante pueden reducir ambigüedades en el proceso de enriquecimiento semántico. Por otra parte, la evaluación del rendimiento demostró que el repositorio presentó un bajo consumo de tiempo (10-40ms) y por lo tanto si se aplica como una fase de pre-correspondencia permite reducir el espacio de búsqueda y el consumo total de tiempo para el entorno BeMantics en general, el cual presentó un rendimiento muy pobre.

Palabras Clave: Proceso de Negocio, Semántica del Comportamiento, Isomorfismo de sub-grafos, Patrones de Control de Flujo, Repositorio de Procesos de Negocio.

Structured Abstract

Background: Retrieve business process (BP) has become a critical field in many areas due to the complexity to find services that match exactly the user's requirements. Therefore it may incur in unnecessary resources consumption in order to find adequate BP. In this way, to develop new intuitive, endowed with artificial intelligence and semantic-based BP retrieval methods which can recognize what the user really needs, could speed up the deployment and configuration of new BP; and become an important area of R & D

Goals: Define methods for business process discovery considering search requirements which can take into account the behavioral semantics.

Methods: This work proposes a BP retrieval environment called BeMantics, which is composed of two main modules: a repository with pre-matching methods based on behavioral semantics; and a structural and semantics matching approach which refines the repository results using an error correction algorithm. This algorithm executes edit operation on a BP in order to make it as similar as possible to other BP called Query BP. To analyze the BeMantics generated results; a pertinence evaluation web tool was designed which allowed a human-judges group to emit relevance judgments on a set of 100 BP. These relevance judgments allow obtaining a relevant BP sub-set, i.e. those BP which would be expected to be recovered by an automatic retrieval tool according to a BP query

Results: A pertinence evaluation model for BP recovery; a behavioral semantics BP repository, a structural and semantic matching analyzer for BPs; a set of 100 BP from geo-processing and telecommunications domains; and control-flow patterns ontology.

Conclusions: BeMantics shows that using semantics in BP tasks comparison can improve slightly the matching precision, but, using very specific domain ontologies can limit the semantics abilities respect to generic domain ontologies; however those can reduce ambiguities in the semantic enrichment process. Furthermore, performance evaluation showed repository presented low-time consumption (10-40ms) and therefore if it is applied as a pre-matching phase can reduce the search space and the total time for the BeMantics environment which sowed very poor performance.

Keywords: Business Process, Behavioral Semantics, Sub-graph Isomorphism, Control-Flow Patterns, Business Process Repository.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

Lista de Siglas

AC: Acceptable Cost.
aFSA: annotated Finite state automata.
AGNE: Agglomerative Nesting Algorithm.
BeMantics: Behavioral Semantics Business Process Discovery approach.
BeMatch: A Platform for Matchmaking Service Behavior Models.
BP: Business Process.
BPDM: Business Process Definition Meta-Model.
BPEL: Business Process Execution Language.
BPM: Business Process Modeling Language.
BPMI: Business Process Management Initiative.
BPML: Business Process Modeling Language.
BPMN: Business Process Modeling Notation.
BPMN: Business Process Modeling Notation.
BPMO: Business Process Modeling Ontology.
BPMS: Business Process Modeling Suite.
CRUD: Create, Read, Update y Delete.
DAML-S: DARPA Agent Markup Language for Services.
DARPA: Defense Advanced Research Projects Agency.
DFS: Depth First Search.
ebXML: Electronic Business using eXtensible Markup Language.
EMF: Eclipse Media Framework
EPC: Event Driven Process Chains.
FSA: Finite State Automata.
ISM: Input Similarity Matrix.
IT: Information Technologies.
jUDDI: java Universal Description, Discovery, and Integration.

LNE: List of Nodes and Edges.
LUCAS: Layer for UDDI Compatibility with Annotated Semantics.
m3pe: Multi Meta Model Process Engineering.
MOF: Meta Object Facility.
NOGOSS: New Generation Operations Systems and Software
OIL: Ontology Interface Layer.
OMG: Object Management Group.
OWL: Web Ontology Language.
OWL-S: Web Ontology Language for Services.
PSL: Property Specification Language.
RDBMS: Relational DataBase Management System.
RDF: Resource Description Framework.
REST: Representational State Transfer.
SAWSDL: Semantic Annotations for WSDL.
SemBiz: Semantic Business Process Management for flexible dynamic value chains.
seTOM: semantic enhanced Telecom Operations Map.
SOA: Service Oriented Architecture.
SOAP: Simple Object Access Protocol.
SPARQL: Simple Protocol and RDF Query Language.
sSID: semantic Shared Information and Data model
SUPER: Semantics utilized for Process Management within and between Enterprises.
TDA: Trace Detection Algorithm.
TIC: Tecnologías de la Información y las Comunicaciones.
TLC: Tratado de libre comercio.
TMF: TeleManagemente Forum
UDDI: Universal Description, Discovery, and Integration.
UML: Unified Modeling Language.
VSM: Vector Space Model
W3C: World Wide Web Consortium
WfMC: Workflow Management Coalition
WS: Web Services
WSDL: Web Service Description Language.
WSMO: Web Service Modeling Ontology.
WSMO4J: WSMO for Java.
WSMT: Web Service Modeling Toolkit

XML: Extensible Markup Language.
XPDL: XML Process Definition Language.
YATOSP: Yet Another Telecoms Ontology, Service and Process.
YAWL: Yet Another Workflow Language.

Capítulo 1

1 Introducción

1.1 Contexto General

En la actualidad las organizaciones IT están buscando nuevas estrategias para construir y desplegar soluciones flexibles y orientadas al servicio, con el fin de responder de manera rápida y con una buena relación costo/beneficio a las condiciones dinámicas del mercado. Una de esas estrategias es adoptar tecnologías flexibles basadas en SOA (Service oriented Architecture) las cuales ofrecen capacidades para la composición dinámica y fácil reúso de componentes software a través de estándares abiertos [1].

Dichos componentes software pueden ser servicios Web (WS) y procesos de negocio (BP). Los WS son unidades software accesibles a través de protocolos estándares de internet; y los BP son servicios complejos que pueden integrar otros componentes software (WS o aplicaciones legadas con interfaces estándares) utilizando un conjunto de tareas interconectadas con propósito de reunir sus funcionalidades individuales y lograr un objetivo de negocio común [2].

En este sentido, SOA ha ganado una gran aceptación dentro de las compañías IT y la comunidad de investigación las cuales han promovido la proliferación de lenguajes, herramientas y componentes software reutilizables [3, 4]. Sin embargo uno de los retos en este contexto es recuperar componentes dentro de grandes

repositorios generados como consecuencia de la proliferación de componentes software. La recuperación de los componentes recuperados debe cumplir características de fácil reúso y bajo tiempo al mercado (time-to-market) de acuerdo a los requisitos de los usuarios.

1.2 Escenarios de Motivación

El mundo se encuentra en constantes transiciones tecnológicas y sociales que afectan enormemente a las economías de los países, especialmente a aquellos que están en vía de desarrollo. Este es el caso de Colombia, un país que busca avanzar a pasos agigantados en su afán por reducir la brecha tecnológica respecto a los países desarrollados. Lo anterior sumado a los nuevos cambios inducidos por tendencias como el TLC (Tratado de Libre Comercio) y la globalización obligan a las empresas nacionales a incrementar su competitividad para ocupar un lugar privilegiado en el mercado buscando satisfacer las necesidades de los clientes modernos [5].

En la situación se puede encontrar empresas prestadoras de servicios en internet (aplicaciones o componentes software), las cuales necesitan crear, adaptar, modificar o integrar servicios existentes con rapidez y fiabilidad. Para alcanzar este objetivo una de las estrategias llevada a cabo por algunas empresas es la reingeniería sobre las TIC, con el fin de desplegar rápida y eficientemente nuevos servicios de valor agregado, adoptando un paradigma orientado al servicio que proporciona la racionalización de su infraestructura IT [6]. Sin embargo, lograr este objetivo no es fácil ya que se requiere exponer servicios a través de interfaces estándares, como los servicios Web (WS) y los procesos de negocio (BP), con el fin de organizarlos en bloques reutilizables con características de bajo acoplamiento y fácil integración.

Otro escenario dónde se puede presentar esta situación, es el caso de las integraciones horizontal o vertical de organizaciones, lo cual se produce como resultado de los rápidos cambios en las economías y tendencias mundiales y las

consecuentes modificaciones en los modelos de negocio de las empresas. La integración organizacional se consolida en alianzas, fusiones, adquisiciones y convenios entre las organizaciones [7-9], como parte del comportamiento de las empresas que buscan su expansión en los nichos de mercado para controlar y evaluar costos, precios, producción y rentas [10]. Este fenómeno resulta fundamental en la región latinoamericana para afrontar con garantías el futuro tan cambiante del mercado [11].

Como resultado, la integración de las organizaciones requiere de una reestructuración de los BP internos, lo cual constituye todo un reto dado que cada organización maneja los procesos de diferente manera. Un ejemplo de esto son las grandes compañías, la cuales utilizan un gran número de aplicaciones empresariales independientes, y hacer un cambio en cualquiera de ellas requeriría el análisis y re-codificación de miles de interfaces [12]. Lo anterior pone en evidencia la necesidad de reorganizar los procesos internos de cada organización con el fin de que puedan interoperar y cooperar con nuevos procesos, que seguramente estarán presentes después de realizar la integración organizacional.

Tanto en el primer como en el segundo escenario aparece la reorganización de BP como un factor fundamental, sin embargo se requiere del uso de modelos que faciliten la búsqueda de procesos existentes que se asemejen lo suficiente al BP solicitado por el cliente, de tal manera que la integración sea rápida y tenga un menor impacto sobre los procesos que normalmente se ejecutaban dentro de la organización.

Un ejemplo de empresas que padece este problema se puede encontrar en el sector de las telecomunicaciones, el cual está experimentando los cambios más rápidos e impredecibles del mercado, que sumados a la aparición de nuevas tecnologías exigen constantes ajustes en sus modelos de negocio. Además los clientes actuales ya no se conforman con los simples servicios de voz o datos, sino que demandan por servicios más avanzados e integrados [13]. Para esto dichas empresas deben retomar los servicios existentes, complementarlos y en algunos casos unirlos para crear nuevos BP más avanzados y adaptables a las necesidades de sus clientes.

1.3 Definición del Problema

En la actualidad existe una variedad de métodos de recuperación de componentes software, los cuales se pueden clasificar principalmente en cuatro niveles de descubrimiento: interfaces, semántica, estructura y comportamiento.

El nivel de interfaces, permite buscar palabras clave relacionadas con cadenas de texto asociadas a las entradas, salidas o nombres de los servicios o tareas de los componentes software [14, 15]. El nivel de semántica, utiliza ontologías de dominio para inferir sobre un conjunto de conceptos relacionados con los nombres, entradas, salidas y tipos de los componentes software [1, 16-20]. El nivel estructural, compara componentes software estructurados, como por ejemplo composiciones de WS a través de BP, para lo cual se utilizan algoritmos de isomorfismo de grafos [21-23]. El nivel de comportamiento compara dos componentes software estructurados utilizando criterios basados en el control de flujo (es decir constructores específicos que definen como se comporta un BP) [24-26], o criterios basados en los registros de ejecución de los BP [27-29].

Estos cuatro niveles, generalmente han sido abordados por otros trabajos de investigación de manera separada, es decir que sólo utilizan uno de los niveles a la vez. No obstante, se ha demostrado que en muchas situaciones la aplicación de estos métodos de manera independiente no es suficiente ya que se requiere de la combinación de ellos para lograr resultados que se adapten a las exigencias de los usuarios [30-32] y el uso de técnicas de indexación que permitan acelerar el proceso de descubrimiento.

Finalmente, de acuerdo a todas las consideraciones descritas en esta sección el presente proyecto plantea la siguiente pregunta de investigación:

¿Cómo reducir los tiempos de despliegue de BP considerando una fase de descubrimiento que se adapte a los requerimientos de búsqueda en cuanto a semántica del comportamiento?

1.4 Antecedentes

La investigación descrita en este trabajo de grado parte de las bases cimentadas por el trabajo doctoral del Dr. Corrales [33], en el cual se plantea el descubrimiento de BP teniendo en cuenta su estructura y relaciones léxicas de sus tareas componentes. El aporte principal del trabajo del Dr. Corrales radica en su técnica de correspondencia, la cual no solo considera equivalencias exactas entre los procesos comparados, sino que también tiene en cuenta correspondencias aproximadas utilizando operaciones de edición (eliminación o sustitución) de nodos y aristas dentro de un grafo que representa al BP. Sin embargo, esta técnica solo considera un nivel estructural y un nivel léxico simple fundamentado principalmente en la base de datos léxica conocida como WordNet [34].

En la presente propuesta se complementa el trabajo del Dr. Corrales, a través de dos fases de descubrimiento; la primera denominada fase de pre-correspondencia la cual crea un índice basado en la *"semántica del comportamiento"*, y no solo tiene en cuenta aspectos estructurales sino que presta especial atención al comportamiento en cuanto a la semántica del flujo de ejecución (control de flujo); a segunda, denominada fase de correspondencia, funciona de manera similar al trabajo del Dr. Corrales en cuanto a la correspondencia estructural de los BP, pero en lugar de comparación léxica de tareas utiliza un enfoque semántico en un dominio específico de aplicación en el contexto de las telecomunicaciones.

1.5 Alcance

El presente trabajo de grado estudió las técnicas de descubrimiento en los cuatro niveles descritos en la sección 1.3 y con base en ellos propuso un mecanismo divido en dos fases; la primera denominada de pre-correspondencia basada en semántica del comportamiento, es decir, semántica aplicada al control de flujo; y la segunda denominada correspondencia, la cual es una técnica multinivel que básicamente integra los niveles de interfaces, semántica y estructura.

Adicionalmente, se estudiaron y diseñaron 100 BP de los entornos de telecomunicaciones y geo-procesamiento con el fin de construir una base de prueba para analizar el rendimiento y relevancia del mecanismo propuesto. En este aspecto, cabe aclarar que solamente aquellos BP del dominio de las telecomunicaciones fueron semánticamente enriquecidos en tanto a los tipos de datos de sus interfaces (entradas y salidas) y los nombres de sus tareas.

En cuanto al tiempo de despliegue de servicios se toman como referencia los trabajos de Nokia Siemens Networks e IBM [35], en el cual se demuestra que reutilizar servicios permite obtener una reducción del tiempo usual de despliegue de los mismos, desde seis meses a tan solo dos semanas y media; y el trabajo de Ramírez y Rojas [36] en el cual se propone la reducción del mismo tiempo a tan solo diez días.

De acuerdo a lo anterior y teniendo en cuenta que el descubrimiento es una fase fundamental de la composición y la reutilización de servicios y procesos de negocio, se puede afirmar que los mecanismos de descubrimiento automáticos permiten acelerar el despliegue y configuración de nuevos BP y servicios reutilizando componentes recuperados.

1.6 Contribuciones y Resultados Principales

- Estudio del estado actual del conocimiento en lenguajes de modelado de BP.
- Estudio del estado actual del conocimiento en formalismos de modelado de BP.
- Estudio del estado actual del conocimiento en mecanismos de descubrimiento de BP.
- Documentación sobre enriquecimiento semántico de un BP utilizando el lenguaje BPMO.
- Un mecanismo de indexación que implementa la fase de pre-correspondencia basada en semántica del comportamiento.

- Adaptación de dos ontologías del dominio de las telecomunicaciones.
- Un mecanismo de correspondencia que permite comparar estructuralmente dos BP y que implementa un mecanismo de comparación semántica de las tareas de dos BP.
- Una plataforma de evaluación de la pertinencia de herramientas de recuperación de BPs.
- Artículo “Plataforma para Evaluar Sistemas de Recuperación de BP” presentado en la revista de Investigaciones UCM la cual está indexada en categoría C del sistema publindex de Colciencias. - 2011
- Artículo “Business Process Model Retrieval Based on Graph Indexing Method” publicado en el journal *Business Process Management Workshop* de Springer Berlin Heidelberg y aceptado también dentro del workshop “rBPM – reuse in Business Process Management” en el marco de la conferencia “Business Process Management” en Hoboken New Jersey, septiembre de 2010.
- Artículo “Business Process Repository based on Control Flow Patterns” presentado en la quinta conferencia euroamericana de sistemas telemáticos e informáticos (EATIS), en Panamá – Septiembre de 2010.
- Artículo “Comparación Semántica de Tareas entre BP de Telecomunicaciones” presentado en el 5 Seminario Nacional de Tecnologías Emergentes en Telecomunicaciones y Telemática, Popayán – Junio - 2010
- Codirección en la Monografía de pregrado “BÚSQUEDA SEMÁNTICA EN UN REPOSITORIO DE PROCESOS DE NEGOCIO” realizada por los ingenieros David Corchuelo y Daniel Rivas.
- Codirección en la Monografía de pregrado “COMPARACIÓN SEMÁNTICA DE TAREAS ENTRE DOS PROCESOS DE NEGOCIO DE TELECOMUNICACIONES” realizada por los ingenieros Adriana Bastidas y Leandro Ordóñez.
- Codirección en la Monografía de pregrado “PLATAFORMA PARA LA EVALUACIÓN DE SISTEMAS DE RECUPERACIÓN DE SERVICIOS BASADOS EN COMPORTAMIENTO” realizada por la ingeniero Laura Sandino.
- Informe de pasantía realizada en la Universidad Politécnica de Turín (Turín - Italia)
- Informe de actividades de docencia ejecutadas entre el periodo I -2008 al I-2010.

- Un disco compacto que contiene toda la información en formato digital generada en el transcurso del proyecto.
- Documentos anexos que presentan información adicional útil de complemento de esta monografía.

1.7 Estructura de la Monografía

El presente trabajo de grado se estructura de la siguiente manera: En el capítulo uno se presenta la introducción, en la cual se hace una corta contextualización acerca de antecedentes, problema central y alcance del proyecto. En el capítulo dos se estudian los conceptos principales y los trabajos relacionados con la temática de recuperación de procesos de negocio. El capítulo tres, describe el módulo de pre-correspondencia, el cual implementa el mecanismo de almacenamiento y primer módulo de recuperación de procesos de negocio. El capítulo cuatro, presenta al módulo de correspondencia, el cual refina los resultados del módulo de pre-correspondencia, ejecutando un análisis estructural y semántico basado en un algoritmo de corrección de errores. El capítulo cinco, discute el entorno para la evaluación experimental de la propuesta y los resultados conseguidos. Finalmente en el capítulo seis se presentan las principales conclusiones obtenidas por el presente trabajo.

Capítulo 2

2 Estado Actual del Conocimiento

Este capítulo describe el estado actual del conocimiento alrededor de la recuperación de BP. Inicialmente en la sección 2.1 se presenta la base conceptual sobre la cual se centra la presente propuesta y a continuación en la sección 2.2 se exponen los principales trabajos relacionados.

2.1 Base Conceptual

2.1.1 Arquitectura Orientada al Servicio (SOA).

SOA es un paradigma arquitectónico que ha ganado una importante atención dentro de las tecnologías de la información (TI) y las comunidades de negocios [37]. Este paradigma representa una evolución de la computación distribuida que permite la composición dinámica y fácil reúso de componentes software a través de estándares abiertos [1].Por esta razón, se ha convertido en una tendencia fundamental para el diseño, desarrollo e integración de nuevas aplicaciones software través de la orientación al servicio [38, 39], la cual permite integrar negocios utilizando tareas de negocio representadas por componentes software modulares, reutilizables y ampliamente escalables.

Dichos componentes software se conocen como servicios complejos y pueden estar constituidos por otros servicios más simples con el propósito de reunir sus funcionalidades para cumplir un fin de negocio común [40, 41]. La composición de estos servicios se puede realizar a través de dos mecanismos principales; el primer mecanismo se conoce como orquestación de servicios, en este caso la composición se estructura dentro de un proceso central que coordina un grupo de servicios que no conocen el proceso del cual forman parte ni el objetivo de la composición, es decir, existe solo un ente central que coordina y conoce el objetivo de negocio de la composición: en contraste, en el segundo mecanismo conocido como coreografía de servicios, no existe un coordinador central, sino que cada servicio involucrado conoce exactamente cuándo tiene que ejecutar las operaciones que debe realizar y los socios con los que debe interactuar. En la coreografía todos los servicios necesitan conocer, el siguiente servicio a ejecutar, las operaciones, los mensajes a intercambiar y el tiempo de ejecución [42].

2.1.2 Ontología

Ontología es un concepto tomado de la filosofía y aplicado a la informática que permite realizar inferencias sobre un dominio del conocimiento. En la informática específicamente se ha convertido en una de las herramientas más importantes de las Web Semántica, ya que de acuerdo con la W3C (World Wide Web Consortium) permite “definir formalmente un conjunto común de términos que se utilizan para describir y representar un dominio” [15]. En este sentido el uso de ontologías brinda el acceso a mecanismos que permiten dotar a los sistemas de búsqueda de información con algo de inteligencia artificial, a través de inferencia de conceptos sobre un dominio de conocimiento preestablecido.

En el campo de los BP y los WS, las ontologías son útiles para encontrar correspondencias entre las tareas de los BP o las operaciones de los WS de acuerdo a las relaciones entre los conceptos que las describen. Los principales estándares de ontologías son RDF (Resource Description Framework) [43], DAML (DARPA Agent Markup Language)[44], OIL (Ontology Interface Layer)[45] y OWL (Web Ontology Language)[46].

2.1.3 Servicios Web (WS)

Los WS son representaciones lógicas de actividades de negocio que tienen resultados específicos; son auto-contenidos, pueden estar compuestos por otros servicios y constituyen cajas negras para los consumidores del servicio [39]. Además, pueden estar descritos, publicados, localizados e invocados en la red a través de estándares abiertos de internet, y por lo tanto facilitan la interoperabilidad entre distintas aplicaciones [47].

Actualmente existen dos métodos principales para utilizar y describir WS en internet; el primero es el lenguaje de descripción de WS (WSDL por sus siglas en inglés)[48], el cual está basado en XML(Extensible Markup Language) y permite describir servicios basados en el protocolo SOAP (Simple Object Access Protocol); y el segundo conocido como RESTFul, el cual permite implementar servicios utilizando HTTP y los principios del estilo arquitectónico REST (Representational State Transfer) inicialmente definido por Fielding y Taylor [49].

2.1.4 Procesos de Negocio (BP)

Los procesos de negocio (BP por sus siglas en inglés) están relacionados con el concepto de orquestación, ya que permiten organizar componentes software dentro de una estructura de ejecución centralizada que gestiona el orden en que los componentes se utilizan. Dichas estructuras de ejecución se pueden entender como servicios complejos con funciones suministradas por diferentes servicios, los cuales ya existen en la Web y están dinámicamente integrados para garantizar una tarea de negocio más compleja [50].

De acuerdo con la WfMC (Workflow Management Coalition) un BP es un conjunto de uno o más procedimientos o actividades que colectivamente alcanzan un objetivo de negocio o unas políticas, dentro del contexto de una estructura organizacional, definiendo roles y relaciones funcionales [51]. En este sentido los BP se pueden modelar a través de la captura de secuencias ordenadas de actividades e

información de apoyo que una empresa realiza para conseguir sus objetivos de negocio [52].

En la actualidad los BP se pueden modelar utilizando diferentes modelos de definición a través de lenguajes y formalismos de modelado de BP. En las secciones 2.1.6 y 2.1.7, se hace un estudio de lenguajes y formalismos; y se selecciona uno en cada categoría para representar los BP en la presente propuesta.

2.1.5 Patrones de Control de Flujo

El control de flujo es una característica de los BP que está estrechamente relacionado con el comportamiento en cuanto a los flujos de ejecución que pueden tomar lugar dentro de un BP (de acuerdo a nodos de control como por ejemplo XOR (Split-Join), AND (Split-Join) y OR (Split-Join), en otras palabras las relaciones de dependencia (paralelismo, sincronización, selección entre otras) entre sus tareas.

En este sentido los patrones de control de flujo corresponden a estructuras que representan comportamientos específicos de los BP, los cuales fueron inicialmente introducidos por la comunidad investigativa liderada por el Dr. Van der Aalst [53] quien inicialmente propuso 20 patrones de control de flujo, los cuales fueron incrementados posteriormente a 43 en el estudio de Rusell et al [54].

Para mayor comprensión acerca de los patrones de control de flujo se puede encontrar una descripción detallada de los mismos basada en el estudio de Rusell et al en el Anexo A.

2.1.6 Lenguajes de Modelado de Procesos de Negocio

Esta sección presenta una clasificación de los lenguajes de modelado de BP de acuerdo a trabajo desarrollado por Ko, et.al [55], el cual organiza los estándares BPM (Business Process Modeling Language) utilizando el ciclo de vida de BPM. Tal y como se puede observar en la Figura 2-1, este ciclo de vida está compuesto por las

fases de diseño del BP, configuración del sistema, despliegue del proceso y diagnóstico.

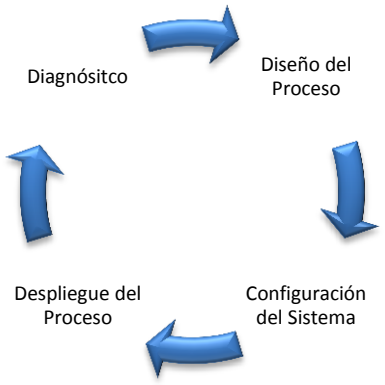


Figura 2-1: ciclo de vida de BPM
Figura tomada de [56] citado por [55]

De esta manera el trabajo de Ko, et al clasifica los estándares BPM en estándares gráficos, de ejecución, de intercambio y de diagnóstico. Teniendo en cuenta esta clasificación, en el presente trabajo se estudian los lenguajes de modelado de BP en lenguajes gráficos, de intercambio y semánticos.

Lenguajes Gráficos: permiten modelar los BP como diagramas gráficos con transiciones, tareas y flujos de control. En esta categoría se pueden encontrar los siguientes lenguajes:

- **Diagramas de Actividad de UML (UML-D):** los diagramas de actividad corresponden a una herramienta del estándar UML (Unified Modeling Language) del grupo OMG (Object Management Group), el cual en la actualidad es ampliamente aceptado por la comunidad software, específicamente en el campo de la orientación a objetos. Los diagramas de actividad están basados en técnicas de diagramas de flujos y máquinas de estados cuyas actividades son estados y los enlaces entre ellas son transiciones. Entre sus principales características está el soporte para envío y recepción de señales a nivel conceptual; el uso de estados de espera y procesamiento; y la capacidad de descomposición de actividades en sub-actividades. No obstante, posee algunos inconvenientes en cuanto a la captura completa de clases importantes de invocación y eventos diferidos.

- **Cadenas de Procesos dirigidas por Eventos (EPC):** el lenguaje gráfico denominado cadenas de procesos dirigidas por eventos fue introducido por Keller, Nüttgens y Scheer en 1992 dentro del entorno de la arquitectura del Sistema Integrado de Información ARIS[57]. Este lenguaje se centra en la descripción de procesos a nivel de la lógica de negocio (no necesariamente en un nivel formal de especificación) en una manera fácil de comprender y utilizar por profesionales de gestión de negocios. Este lenguaje hereda su nombre de los tipos de diagramas que muestran la estructura del control de flujo de los procesos como una cadena de eventos y funciones, es decir, una cadena de procesos dirigida por eventos [58]. Algunas de las ventajas de EPC es que cubre aspectos y descripciones detalladas de las unidades de organización de negocio junto con sus respectivas funciones y recursos materiales; y que posee un lenguaje rico en el modelado de dominio, de organización, de perspectivas funcionales y de comportamiento. A pesar de estas ventajas, el lenguaje EPC también posee algunas debilidades como por ejemplo la carencia de una sintaxis y semántica bien definidas (para esto generalmente requiere de mapeos a redes de Petri). Además, las perspectivas organizacional y de información están parcialmente soportadas.

- **Notación para el Modelado de Procesos de Negocio (BPMN):** BPMN es un estándar de modelado gráfico de BP creado inicialmente por la BPMI (Business Process Management Initiative) y actualmente mantenido por el grupo OMG. Este estándar permite modelar flujos de BP y WS a través de la coordinación de secuencias de procesos y los mensajes que fluyen entre los participantes de las diferentes actividades o tareas [59]. Una característica importante de BPMN es que se consolida como una notación entendible por diferentes tipos de usuarios, desde analistas de negocios (encargados de crear los borradores iniciales), hasta los desarrolladores de procesos ejecutables (responsables de crear una aplicación que ejecute las tareas del proceso). Otra característica importante de esta notación es que permite que los lenguajes diseñados en XML para la ejecución de BP, tales como BPEL4WS (Business Process Execution Language for Web Services) y BPML (Business Process Modeling Language) se puedan expresar visualmente mediante una notación estándar [59]. En consecuencia, BPMN es un lenguaje de modelado altamente genérico que permite la

transformación entre múltiples lenguajes de especificación ejecutables como BPEL o BPML y además ofrece una interfaz gráfica muy rica para la definición del control de flujo. No obstante, posee algunas desventajas como su capacidad para describir el contexto organizacional y la carencia de meta-modelos que permitan reducir ambigüedades y confusiones al compartir modelos BPMN.

Lenguajes de Ejecución: Esta clase de lenguajes permiten modelar BP ejecutables que se pueden desplegar en un motor de ejecución, el cual ejecuta cada una de sus tareas de acuerdo a unas reglas de control de flujo predefinidas..

- **Lenguaje de Modelado de Procesos de Negocio (BPML):** BPML es un lenguaje basado en XML, que describe la estructura de los procesos y la semántica de su ejecución. Por lo tanto facilita el despliegue de BP en motores que los ejecutan elemento por elemento de acuerdo con la semántica predefinida. Los procesos BPML se basan en estructuras de grafos, con ciclos y caminos paralelos; y constructores de bloques, tales como variables, bloques recursivos y manipuladores de excepciones. Entre las características más importantes de BPML se tiene su capacidad para permitir que los programadores se centren en la definición del proceso y sus secuencias de ejecución (control de flujo) sin preocuparse por lenguajes de bajo nivel; y su facilidad de reusabilidad y escalabilidad en los sistemas de gestión de BP (BPMS). En cuanto a sus debilidades se puede nombrar que BPML no permite evidenciar fácilmente los componentes temporales de un proceso; su baja aceptación en el mercado y que a pesar de ser un estándar en la actualidad se encuentra obsoleto [60].

- **Lenguaje de Ejecución de Procesos de Negocio (BPEL):** es un lenguaje de modelado de BP ejecutables basado en XML y estandarizado por OASIS [61]. Este lenguaje permite la especificación, ejecución, y descripción de WS orquestados a través de BP con capacidades de exportar e importar funcionalidades mediante interfaces estándares conocidas como WSDL. BPEL soporta el modelado de dos tipos de procesos: Un proceso Abstracto que representa un protocolo de negocio para especificar el intercambio de mensajes entre diferentes partes desde la perspectiva de una sola organización y sin revelar su comportamiento interno [62]. Un proceso ejecutable, especifica el orden de ejecución entre un número de actividades que lo constituyen, las partes que están

involucradas, los intercambios de mensajes entre estas partes y los mecanismos de manejo de fallos y excepciones. En cuanto a estructura, un proceso BPEL se divide en cuatro secciones: definición de relaciones con los socios externos (los clientes que utilizan el BP y los WS a los que invoca el proceso); definición de variables; definición de los distintos tipos de manejadores (manejadores de fallos y de eventos), y descripción del comportamiento de los BP [63].

- **Otro Lenguaje de Workflows YAWL:** YAWL fue desarrollado a partir de la definición de patrones de control de flujo introducidos por Rusell et al [54] representados a través de redes de Petri. Dichos patrones especifican diferentes comportamientos en cuanto a ejecución que un BP puede realizar en su ciclo de vida, por esta razón YAWL fue creado para soportar el mayor número posible de dichos comportamientos. Entre las características más relevantes de este lenguaje se encuentran: la adaptación dinámica de modelos de workflows a través de nociones de worklets (objetos que representan un conjunto de tareas); y su modelo basado en XML para la definición y manipulación de datos basado en XML Schema, XPath y XQuery. No obstante, a pesar de sus excelentes características se reduce a constituirse como un esfuerzo académico y como tal no es ampliamente aceptado por la industria.

Lenguajes de Intercambio: son lenguajes de modelado que permiten realizar portabilidad de información entre BP creados en diferentes lenguajes modelado.

- **Metamodelo de Definición de Procesos (BPDM):** BPDM es un estándar definido por el grupo OMG, que ofrece un entorno para representar modelos de BP de manera independiente de la notación o metodología a través de un meta-modelo y un vocabulario de procesos compartido con conexiones bien definidas entre términos y conceptos. Dicho meta-modelo está basado en el estándar MOF (Meta Object Facility) el cual permite capturar los BP y ofrecer una sintaxis XML para almacenar y transferir modelos de BP entre las notaciones y tecnologías facilitando su integración. Por esta razón BPDM actúa como un estándar traductor multilenguaje y facilita una abstracción de los elementos básicos y comunes entre lenguajes como BPEL, BPMN, XPDL, XLANG, WSFL y UML-AD [55]

- **Lenguaje XML de Definición de Procesos (XPDL):** XPDL es un lenguaje estándar basado en XML que permite definir BP que soporta todos los aspectos de los procesos BPMN incluyendo las descripciones gráficas y las propiedades ejecutables[64]. Este lenguaje está basado en XML es ampliamente aceptado y fue inicialmente propuesto por la WfMC en su modelo de referencia [65]. Entre sus principales características de puede destacar que es altamente estandarizado y aceptado tanto por la comunidad académica como por la industria; y que posee grandes facilidades de conversión entre diferentes lenguajes gracias a estar fielmente basado en la notación BPMN.

Lenguajes Semánticos: son lenguajes de modelado de BP que permiten la inclusión de anotaciones semánticas para enriquecer características de los procesos, como por ejemplo, los nombres y las interfaces (entradas y salidas) de las tareas.

- **Entorno Semántico para Servicios Web (OWL-S):** OWL-S es un protocolo de modelado de WS semánticos basado en ontologías OWL [66], el cual facilita que los usuarios y los agentes de software puedan descubrir, invocar, componer y vigilar los recursos Web con un alto grado de automatización. Para esto OWL-S brinda características importantes que ayudan al descubrimiento, invocación, composición e interoperabilidad automática de WS utilizando tres conceptos fundamentales: un perfil de usuario, que permite identificar usuarios como consumidores, proveedores y componentes de infraestructura; un modelo de servicio que describe las interacciones entre el cliente y el servicio; y un fondo de servicio (*grounding*) que especifica los detalles de acceso al servicio [67].

- **Ontología para Modelar Procesos de Negocio (BPMO):** BPMO es un lenguaje semántico que permite definir un modelo preciso de descripción semántica para BP. De esta manera BPMO enriquece la descripción de BP en un alto nivel de abstracción para facilitar que los lenguajes gráficos de modelado se puedan interconectar con especificaciones técnicas de implementación [68]. Además implementa una ontología capaz de representar artefactos de varias metodologías de modelado de los BP a través de una representación única y unificada. Dicha ontología se creó dentro del proyecto de la unión europea denominado SUPER[1]

[1] El proyecto SUPER pretendía llevar la Gestión de Procesos de Negocio (BPM) desde el nivel de las tecnologías de la Información (TI) hacia el nivel de negocio. El resultado fue el desarrollo de herramientas que permiten el despliegue de la

(Semantics Utilized for Process Management within and between Enterprises), con el objetivo de unificar dos métodos de modelado de BP, uno basado en grafos y otro basado en bloques. El primero preferido por los usuarios de negocio (para modelar gráficamente los BP) y el último necesario para la traducción a lenguajes ejecutables antes de que la ejecución se realice. Entre las características más importantes de este lenguaje se pueden nombrar: su facilidad de traducción entre diferentes lenguajes; su representación común basada en BPMN en lugar de imponer una nueva notación (como lo hace YAWL), y la facilidad de insertar anotaciones semánticas en las interfaces y nombres de las tareas de los BP. No obstante, no posee gran aceptación debido a que constituye un esfuerzo académico, su documentación es escasa y su soporte se descontinuó con la finalización del proyecto SUPER a finales del año 2009.

2.1.7 Formalismos para el modelado de Procesos de Negocio

En esta sección se presentan los principales formalismos (lenguajes formales) para el modelado de BP. Los formalismos a diferencia de los lenguajes presentados en la sección anterior, que se encargan de modelar aspectos técnicos de los BP, proveen una fundamentación teórica y académica para definir BP sin ambigüedad y con facilidad de análisis matemáticos. A continuación se presentan los principales formalismos para modelar BP.

Autómatas de estados Finitos (FSA): los autómatas de estados finitos (FSA) también conocidos como máquinas de estados finitos (FSM) corresponden a un modelo computacional fundamentado en un estado inicial; un conjunto finito de estados; un alfabeto de entrada; y una función de transición que mapea símbolos de entrada y estados actuales a un estado posterior [69]. Generalmente los FSA se representan a través de grafos dirigidos conocidos como diagramas de transición de estados, en los cuales los nodos representan a los estados y las aristas a las transiciones que los enlazan. En los diagramas de transición de estados las transiciones están etiquetadas con mensajes obtenidos de un conjunto de mensajes y los estados finales están marcados con círculos concéntricos. Formalmente un FSA

Gestión de Procesos de Negocio Semánticos (SBPM). Este proyecto fue financiado por la Unión Europea dentro de las prioridades de la Information Society Technologies (IST). Disponible en Internet: http://www.ip-super.org/

se puede definir como $\mathrm{FSA} = (\Sigma, \mathrm{S}, s_0, \delta, F)$, donde Σ es un alfabeto de entradas, S es un conjunto de estados, s_0 es el estado inicial, δ es la función de transición y F es el conjunto de estados finales (también conocidos como aceptadores).

Por último cabe anotar que los FSA en su forma original no pueden representar semánticas obligatorias de secuencias de mensajes, por lo tanto no tienen semánticas de ejecución paralela como las ofrecidas por lenguajes más expresivos como las Redes de Petri [33]. La Figura 2-2 presenta un ejemplo de un autómata finito cuyas transiciones se denotan con letras minúsculas y sus estados son 1, 2, 3, 4 y 5.

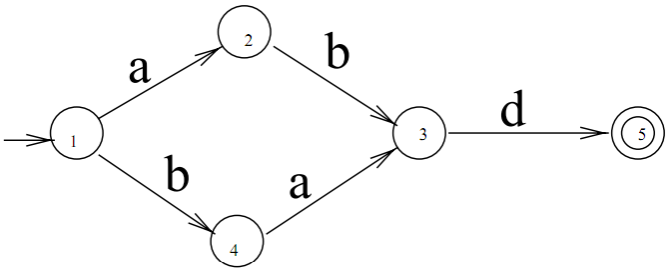


Figura 2-2 Ejemplo de un FSA
Figura tomada de [70]

Redes de Petri (PN): las redes de Petri son herramientas formales para describir y estudiar sistemas concurrentes, asíncronos, distribuidos, paralelos, no deterministas y estocásticos. Una red de Petri se puede ver como un grafo dirigido, bipartito que ofrece una semántica formal para análisis matemáticos y por lo tanto es adecuada para modelar, simular y estudiar BP [68]. Sin embargo, el modelo clásico de las redes de Petri no es apropiado para modelar muchos sistemas que se encuentran en logística, producción, comunicaciones, manufactura flexible y procesamiento de información, por lo cual se hace necesarias algunas extensiones como por ejemplo las redes de workflows y los patrones de workflows [54].

Formalmente una red de Petri se puede definir como $PN = (P, T, A, W, M0)$, donde $P = (p1, p2, \dots, pm)$ es un conjunto finito de lugares, $T = (t1, t2, \dots, tn)$ un conjunto finito de transiciones, $A \subseteq (P \times T) \cup (T \times P)$ el conjunto de aristas

(relación de flujo), $W : A \rightarrow (1,2,3,\ldots)$ una función de peso y $M0 : P \rightarrow (0,1,2,3,\ldots)$ el marcado inicial con $P \cap T = \emptyset$ y $T \cap P = \emptyset$.

El lenguaje matemático de modelado que proveen las Redes de Petri, define un formalismo que permite especificar y describir, tanto la estructura como el comportamiento de sistemas distribuidos de eventos discretos. La expresividad asociada a esta representación formal es mayor que la ofrecida por los formalismos de Grafos y Máquinas de Estado, siendo considerados estos dos últimos, como redes de Petri con sintaxis restringida. Sin embargo, esta potencia expresiva compromete el rendimiento de operaciones de análisis ejecutadas sobre un sistema modelado con esta representación debido a la complejidad computacional de los métodos que hasta la fecha se han desarrollado para implementar dichas operaciones. En la Figura 2-3 se puede observar un ejemplo de una red de Petri.

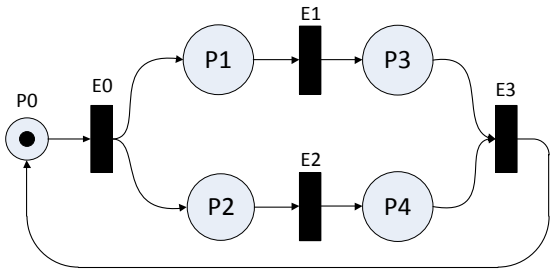


Figura 2-3: Ejemplo de una red de Petri.

Algebra de proceso (Π-Calculus): las álgebras de procesos también conocidas como pi-calculus, fueron introducidas por el matemático Robin Milner en los 90 y corresponden a una familia de métodos para modelar formalmente sistemas concurrentes que ofrecen un entorno de representación, simulación, análisis y verificación de sistemas de comunicación móvil. El lenguaje pi-calculus permite definir procesos concurrentes que interactúan con otros dinámicamente. Cada proceso consiste de una o más acciones (envío o recepción de información en un canal [71]) , las cuales se pueden enlazar en secuencia, en paralelo, en caminos condicionales, o de manera recursiva. En este sentido, algunos estudios han demostrado que el lenguaje pi-calculus está bien adaptado para modelar BP clásicos en cuanto a coreografía y orquestación [68]

La Figura 2-4 muestra un ejemplo de un proceso de reservación de viajes, en el cual un agente de viajes, un cliente y una aerolínea interactúan. El ejemplo está modelado con el lenguaje pi-calculus.

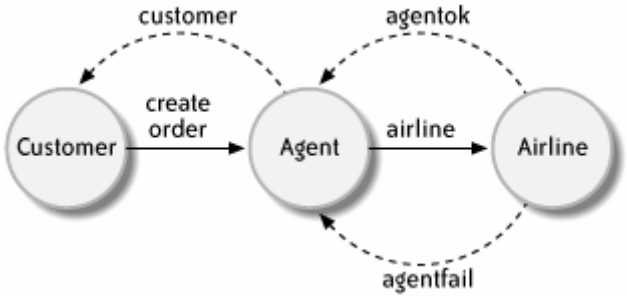


Figura 2-4: ejemplo de un proceso de reservación de viajes representado en Π-calculus

Figura tomada de [71]

Grafos de Proceso: los grafos constituyen una herramienta matemática de modelado formal que utiliza estructuras de datos poderosas y generales para representar objetos y conceptos. Los grafos están constituidos por nodos, que expresan operaciones o cálculos; y aristas o enlaces entre ellos que representan dependencia de datos [72]. De este modo los grafos pueden modelar estructuras con alto nivel de abstracción, y la representación de relaciones de causalidad entre componentes, flujos de datos, control de flujo y concurrencia en la ejecución de procedimientos paralelos. Por esta razón se consideran como excelentes herramientas para modelar BP en los cuales los nodos representan tareas y las aristas el intercambio de mensajes y datos entre los nodos. A este tipo de grafos, que representan BP, se les conoce como *"grafos de procesos"*, en los cuales existen nodos simples, que tienen una arista de entrada y otra de salida; y nodos de control de flujo, que pueden tener múltiples entradas y salidas dependiendo de su tipo (XOR (Split-Join), AND (Split-Join) y OR (Split-Join)). De esta manera los flujos de ejecución de un grafo de proceso pueden ser concurrentes y evitar conflictos. En la Figura 2-5 se puede observar un grafo en el cual el flujo de datos después del nodo *E0* se divide en el nodo de control *XOR-Split* y se sincroniza en el nodo de control *XOR-Join*. Además entre estos nodos están los nodos básicos *E1* y *E2* los cuales toman lugar de manera no determinista.

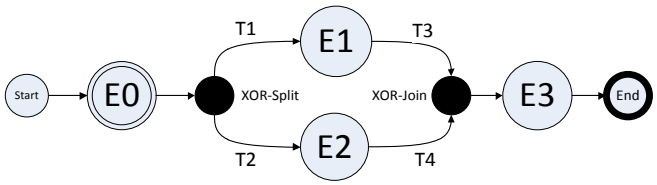


Figura 2-5: Ejemplo de un grafo de proceso

Formalmente un grafo se denota como $G = (V, E)$ donde V es un conjunto (llamado conjunto de vértices o nodos) y E un subconjunto de VxV (conjunto de aristas). A continuación se describen algunos conceptos importantes en la teoría de grafos.

- **Aristas:** las aristas representan la comunicación entre los nodos de un grafo y se pueden expresar a través de un par $\{a, b\}$. En el caso de que la arista carezca de dirección se puede denotar indistintamente $\{a, b\}$ o $\{b, a\}$, siendo a y b los nodos que une, también conocidos como extremos.

- **Nodos:** los nodos son los vértices del grafo que representan cálculos u operaciones de los BP. Se dice que dos nodos v, w son adyacentes si existe una arista $\{v, w\}$. Se denomina grado de un nodo al número de aristas de las que es extremo y dependiendo de su grado se puede denotar al nodo como "par"' o "impar".

- **Caminos:** los caminos representan un conjunto de vértices unidos por una secuencia de aristas. Formalmente se dice que existe un camino en un grafo *G* desde un nodo a hasta un nodo b si existe una sucesión finita no vacía de aristas $\{a, v_1\}, \{v_1, v_2\}, \ldots, \{v_n, b\}$. En este caso se cumplen las siguientes propiedades:
    - Los nodos a y b se denominan *"extremos del camino".*
    - El número de aristas del camino se llama longitud del camino.
    - Si los nodos no se repiten en el camino se dice que es un camino propio o simple.
    - Si hay un camino no simple entre 2 nodos, también habrá un camino simple entre ellos.
    - Cuando los dos extremos de un camino son iguales, el camino se llama circuito o camino cerrado.
    - Un ciclo es denominado un circuito simple

- Un nodo se dice accesible desde el vértice b si existe un camino entre ellos. Todo vértice es accesible respecto a si mismo

Finalmente, entre las propiedades de los grafos como mecanismo formal de modelado de BP se destaca su simplicidad, alto grado de abstracción y la existencia de múltiples herramientas matemáticas para su análisis. Sin embargo, debido al alto grado de abstracción y simplicidad puede presentar algunos problemas de expresividad como por ejemplo la representación de flujos de datos en el cual las redes de Petri resultan más expresivas.

2.2 Trabajos Relacionados

Reutilizar BP es uno de los aspectos más avanzados para el despliegue rápido y eficiente de nuevos productos software en compañías IT. Sin embargo, descubrir aquellos BP con mayor facilidad de adaptación a nuevos productos es todo un reto debido a la proliferación de BP existentes en la actualidad y a las condiciones dinámicas del mercado. En esta sección se realiza un estudio acerca del estado actual de conocimiento alrededor de la temática de descubrimiento de BP y se clasifican las técnicas para este propósito en cuatro niveles: interfaces, semántica, estructura y comportamiento.

2.2.1 Descubrimiento basado en Interfaces

En el nivel de interfaces se tienen en cuenta parámetros relacionados con las entradas y salidas de cada proceso, de las tareas que lo constituyen y de los WS involucrados. En este caso los servicios/procesos pueden estar almacenados en registros como UDDI y ebXML, en los cuales el descubrimiento se puede llevar a cabo a través de las interfaces de descripción WSDL de los servicios. Por lo tanto este nivel está generalmente basado en búsqueda por palabras clave o tablas de correspondencia de parejas. Los principales trabajos en este nivel son los siguientes:

Stroulia y Wang [14] discuten un conjunto de métodos para estimar la similitud de especificaciones de interfaces WSDL para lo cual utilizan una base de datos léxica llamada WordNet [34]. Wordnet permite encontrar relaciones léxicas entre dos cadenas de texto que representan a los identificadores y descripciones de las especificaciones WSDL y la estructura de sus operaciones (mensajes y tipos de datos). La similitud de interfaces utilizada en este trabajo se reduce a una búsqueda de relaciones léxicas y sinonímicas entre los conceptos de las interfaces sin tener en cuenta sus relaciones semánticas.

Kokash, van den Heuvel et al [15] presentan un algoritmo denominado WSDL-M2, el cual combina dos técnicas de correspondencia, la primera denominada léxica para calcular la similitud lingüística entre descripciones de conceptos, y la segunda denominada estructural para evaluar la similitud total entre conceptos compuestos. Al igual que el caso anterior solo tienen en cuenta relaciones léxicas y de estructura en cuando a los conceptos encontrados en las interfaces de los BP y no relaciones de comportamiento o estructura de los procesos.

Xu, Liu et al [73] presentan un modelo de descubrimiento de BP en tres capas: primero la capa de componentes relacionada con información macroscópica de las unidades de negocio y sus interrelaciones; segundo, la capa de operación que captura el conocimiento acerca de las operaciones de negocio realizadas por los participantes; y tercero la capa de integración de operaciones que utiliza un algoritmo denominado *“backtracking”* con el cual analizan dónde las entradas y salidas de las operaciones de negocio pueden coincidir. Finalmente, para analizar sus resultados representan su solución a través de una máquina de estados la cual se compara con los requisitos de un analista de negocios expresados a través del lenguaje PSL (Property Specification Language). El enfoque de este artículo está a un nivel más alto que el técnico, es decir, se centra básicamente en unidades de negocio que están definidas principalmente por analistas de negocio y por lo tanto deja de lado los aspectos técnicos de los procesos como el comportamiento y la estructura.

Koschmider, Hornung et al [74] presentan un editor de modelos de BP el cual utiliza un mecanismo de asistencia al usuario modelador a través de una búsqueda de modelos de negocio o partes de éstos, permitiendo así que el usuario pueda reutilizarlos en la construcción de nuevos BP. Para esto utilizan dos métodos de

búsqueda. El primero denominado búsqueda básica permite indexar BP a través de palabras clave relacionadas con atributos del proceso y calificar los resultados a través de la unión de un modelo de espacio vectorial (VSM por sus siglas en inglés) y un modelo booleano. El segundo método denominado búsqueda extendida tiene en cuenta las variantes de los BP y diferentes criterios de búsqueda que permiten limitar los resultados de la consulta; dichos criterios están basados en el perfil de usuario, la frecuencia de términos y el número de operaciones que se ejecutan en las cadenas de texto que representan a los BP. Sin embargo, su sistema de búsqueda no tiene en cuenta la estructura, el control de flujo (comportamiento), ni tampoco la semántica de las operaciones ya que en este último caso solo utiliza la base de datos léxica Wordnet.

Gonçalves da Silva, Ferreira et al [1] proponen un método denominado DynamiCoS para descubrir correspondencias exactas y parciales entre WS. Para esto realizan extracción de información de entradas, salidas, precondiciones, efectos y metas de los WS y las convierten en anotaciones semánticas que posteriormente se almacenan en un registro denominado jUDDI (jUDDI es un registro mejorado del estándar UDDI). En esta propuesta solo descubren servicios simples, es decir que no se tienen en cuenta composiciones o BP y por lo tanto no consideran aspectos como la estructura y el comportamiento.

Además de los anteriores trabajos, en este nivel se pueden nombrar, los motores de búsqueda de WS en internet, los cuales a partir de palabras clave introducidas por un usuario en una página Web, realizan una exploración dentro de descripciones de WS publicados en diferentes portales con el fin de encontrar aquellos que las contengan. Ejemplos de este tipo de motores de búsqueda son: Web ServiceX[2], Seekda[3], Web Services List[4] y Woogle[5].

Se puede concluir que este nivel de descubrimiento basado en interfaces es interesante dada la facilidad con la que los usuarios pueden encontrar WS con simplemente ingresar una palabra clave. Sin embargo, utilizar solo este nivel como

[2] Disponible en la dirección: http://www.webservicex.net/
[3] Disponible en la dirección: http:// http://webservices.seekda.com/
[4] Disponible en la dirección: http://www.webservicelist.com/
[5] Disponible en la dirección: http://db.cs.washington.edu/woogle.html

mecanismo de descubrimiento es ineficiente debido a que omite detalles importantes acerca de la implementación, comportamiento, estructura y reglas semánticas de los BP de acuerdo a un dominio de aplicación.

2.2.2 Descubrimiento basado en Semántica

El nivel semántico se centra en la capacidad de los sistemas de recuperación de BP y WS para inferir acerca de los conceptos sobre los cuales el cliente está realizando su búsqueda, para esto se utilizan criterios ontológicos que tienen en cuenta el significado y las relaciones entre conceptos. De esta manera en este nivel se busca definir una distancia semántica (valor numérico) entre dos conceptos relacionados con características de las tareas de los BP (como por ejemplo, identificadores o nombres y tipos de datos de entradas/salidas) para determinar si tienen algún tipo de relación conceptual. Los trabajos más relevantes alrededor de este nivel se describen a continuación:

Châtel [75] describe un modelo para la localización semántica de WS y BP, en el cual se consideran los aspectos dinámicos y estáticos de sus descripciones. Dicho modelo adiciona anotaciones semánticas sobre las descripciones de los WS y BP con el fin de relacionarlos con información contenida en ontologías y de esta manera facilitar la publicación y descubrimiento de declaraciones semánticamente enriquecidas en un registro UDDI. Lo anterior, se realiza a través de un marco semántico basado en lenguajes y tecnologías comunes de la semántica como por ejemplo: SAWSDL(Semantic Annotations for WSDL) [76], BPEL, OWL y el servicio de registro UDDI. Además, adiciona un conjunto de librerías (API) denominado LUCAS (Layer for UDDI Compatibility with Annotated Semantics) el cual permite realizar mapeos entre SAWSDL y UDDI. Sin embargo, este trabajo no considera las declaraciones del servicio como un todo sino que se centra únicamente en las operaciones.

Klusch, Fries et al [18] presentan un método de correspondencia híbrida para WS semánticos, el cual complementa el razonamiento lógico con correspondencia aproximada basada en cálculos de similitud sintáctica. Este método está implementado a través de un prototipo denominado OWLS-MX el cual toma un

servicio descrito en el lenguaje OWL-S como consulta y retorna un conjunto ordenado de servicios relevantes que se clasifican de acuerdo con categorías de similitud (exact, plug-in, subsumes, subsumed-by y nearest-neighbor). Lo anterior es posible dado que OWLS-MX utiliza un razonamiento basado en la lógica y las técnicas de recuperación de información de acuerdo al contenido del perfil de interfaces (entradas y salidas) de servicios OWL-S.

Lin y Arpinar [20] presentan una técnica basada en pre y post condiciones de los WS, en la cual las capacidades de los servicios se expresan semánticamente a través de un conjunto de tripletas RDF. Para esto utilizan una ontología que les permite descubrir las relaciones entre dos servicios aun cuando las condiciones no coincidan sintácticamente.

Okkyung y Sangyong [19] presentan un motor de búsqueda de servicios basado en la combinación de reglas con ontologías. Para esto proponen un método que permite entregar resultados personalizados en las búsquedas de WS ya que ejecutan un análisis de varias situaciones posibles y búsquedas mediante un servicio de inferencia y extracción de reglas. Su objetivo es proveer un marco automático y fundamental para la integración de aplicaciones y procesos para pequeñas, medianas y grandes empresas. Posee algunos puntos débiles como la complejidad debida a la gran variedad de algoritmos de activación utilizados, lo cual incrementa el consumo de tiempo en las búsquedas.

Kiefer, Bernstein et al [77] presentan un motor de consultas imprecisas denominado iSPARQL(interactive Simple Protocol and RDF Query Language) el cual utiliza un lenguaje de consultas declarativas con razonamiento estadístico con el fin de simplificar el diseño e implementación de aplicaciones Web. Para evaluar la efectividad de sus resultados hacen un paralelo entre las medidas de similitud realizadas por humanos en un dominio específico en comparación con las ejecutadas de manera automática con inteligencia artificial. El problema principal de este trabajo es que solo actúan en dependencia del dominio y no realizan correspondencia ni alineación de ontologías.

Verma, Akkiraju et al [78] presentan un método para orquestar dinámicamente WS a través de un flujo de BP considerando dependencias inter-servicios y

restricciones. Utilizan un registro UDDI enriquecido semánticamente y un sistema de correspondencia basado en DAML-S que revisa las restricciones y genera un conjunto de servicios compatibles. Una de las principales características de este trabajo es que toma como entrada una representación del flujo del proceso desde el punto de vista de un consultor de negocios (BPM) en lugar de un profesional IT (BPEL).

El nivel de descubrimiento basado en semántica presenta trabajos importantes que permiten relacionar conceptos relativos a los nombres e interfaces de los WS y de las tareas de los BP con conceptos semánticos organizados en una ontología. Sin embargo al igual que en el nivel de interfaces, están generalmente orientados a WS simples y por lo tanto, no tienen en cuenta características importantes de los BP como el comportamiento y la estructura.

2.2.3 Descubrimiento basado en Estructura

Este nivel se centra en la comparación de la estructura de los BP generalmente representados a través de formalismos de modelado. Los cuales son útiles en este nivel ya que facilitan el análisis las estructuras utilizando técnicas matemáticas como por ejemplo el isomorfismo de grafos que permite evidenciar si dos estructuras son iguales. Los principales trabajos en este nivel son:

Corrales, Grigori et al [21] comparan BP utilizando un algoritmo de correspondencia estructural el cual permite buscar grafos de proceso dentro de un repositorio de acuerdo a una consulta representada también como un grafo. Dicho algoritmo también se conoce como algoritmo de corrección de errores debido a que aplica un conjunto de operaciones de edición sobre cada uno de los grafos del repositorio con el fin de hacerlos tan similares como sea posible al grafo de consulta. Además el trabajo de Corrales, Grigori et al utiliza, como apoyo al algoritmo de comparación estructural, un algoritmo de comparación léxica entre cada una de las actividades o tareas de los BP. En este algoritmo léxico se confrontan las etiquetas de los nombres de las tareas y se encuentra una distancia léxica utilizando algoritmos aplicados sobre cadenas de texto que tienen en cuenta abreviaturas, sinónimos y n-gramas. No obstante, a pesar los importantes aportes de este trabajo,

no tiene en cuenta la comparación de BP basada en comportamiento (control de flujo), ni tampoco aplica semántica en la comparación de las actividades del BP, en este caso simplemente utiliza la base de datos léxica Wordnet con la cual comprar los nombres de las actividades.

Eshuis y Grefen [22] establecen mecanismos para comparar BP especificados en el lenguaje BPEL utilizando como criterio a los conjuntos de relaciones estructurales encontradas en sus actividades. Dichas relaciones clasifican la similitud entre dos BP como exacto, cuando los dos BP son idénticos estructuralmente, inexacto cuando los procesos se asemejan pero no son idénticos y plug-in cuando uno de los procesos está contenido en otro.

Wombacher y Li [23] presentan dos medidas importantes para evaluar la similitud de BP. La primera medida es similar a los trabajos presentados en el nivel de interfaces, utiliza una distancia lingüística conocida como distancia Hamming la cual permite calcular el menor número de operaciones de edición de símbolos (substituciones, inserciones y eliminaciones) dentro de las cadenas de texto que identifican a las actividades con el fin de hacerlas similares a las cadenas de texto de las actividades de un proceso de consulta. La segunda medida evalúa una distancia estructural de dos BP representados con el formalismo de grafos dirigidos utilizando la técnica de isomorfismo de grafos. En este caso, de manera similar al trabajo de Corrales et al [21], se calcula una distancia de edición para que un grafo se asemeje tanto como sea posible a un grafo de proceso de consulta.

Sakr y Awad [79] presentan un entorno para consultar y reutilizar modelos de BP basados en grafos a través de un lenguaje de consultas denominado BPMN-Q. Este entorno está constituido por un repositorio de grafos que permite almacenar e indexar grafos en una base de datos RDBMS utilizando un esquema relacional fijo; un editor de consultas basado en la Web; y un procesador estructural de sub-grafos que permite consultar grafos teniendo en cuenta los diferentes tipos de nodos y la complejidad de las aristas. Además, el trabajo de Sakr y Awad utiliza un método de comportamiento basado en patrones y anti-patrones frecuentes de proceso pero no es clara la manera en cómo se aplican en el descubrimiento.

Los trabajos del nivel estructural se centran en la comparación de BP a través de algoritmos estructurales basados generalmente en isomorfismo de grafos, pero no tienen en cuenta el comportamiento basado en el control de flujo o en eventos de ejecución. Por otro lado, ninguno de los trabajos referenciados en este nivel utiliza conceptos semánticos aplicados a los identificadores y tipos de datos de las tareas de los BP.

2.2.4 Descubrimiento basado en Comportamiento

La clasificación de los trabajos en este nivel es más compleja dado que existen varios aspectos que pueden definir el comportamiento de un BP, como por ejemplo el intercambio de mensajes dentro de las actividades, los registros de ejecución histórica y el control de flujo. A continuación se describen los principales trabajos en el nivel de comportamiento.

Wombacher, Mahleko et al [80, 81] presentan un mecanismo de correspondencia en el cual los BP se representan utilizando autómatas de estados finitos enriquecidos aFSA (annotatted Finite state automata). Los aFSA están compuestos por un estado inicial, un estado final, un conjunto de estados (asociados con expresiones lógicas entre mensajes) y un conjunto de transiciones etiquetadas con eventos de negocio. De esta manera, la correspondencia entre dos BP se reduce a comparar dos aFSA, las cuales se consideran similares si existe al menos un camino (secuencias de mensajes) común entre los estados de inicio y fin; y se cumple con un conjunto de mensajes obligatorios y opcionales. Los caminos se representan a través de secuencias de ejecución utilizando listas de n-gramas, en las cuales cada n-grama está relacionada con un estado en particular y una lista de ellas puede representar una abstracción del conjunto de estados localizados en una secuencia de ejecución.

Shen y Su [82] presentan un modelo que asocia mensajes intercambiados entre participantes con actividades ejecutadas dentro de un servicio. Las actividades se catalogan en un perfil descrito utilizando OWL –S; los servicios se modelan a través de autómatas finitos no deterministas y la recuperación de los mismos se realiza recurriendo a un lenguaje de consultas desarrollado para expresar propiedades temporales y semánticas en los comportamientos de servicios.

Yun, Yan et al [83] adoptan el formalismo pi-calculus como herramienta formal para especificar y modelar el comportamiento de WS compuestos. En este sentido, presentan dos métodos de correspondencia de BP; uno basado en parámetros de entrada y salida de cada WS, con sus nombres y tipos; y otro basado en correspondencia de comportamiento, en el cual buscan consistencia entre el orden de ejecución de las acciones de dos servicios compuestos. En general el trabajo de Yun, Yan et al está basado en el nivel de comportamiento de acuerdo al orden de ejecución de las actividades, pero no tiene en cuenta realmente el control de flujo, ni tampoco la semántica de las interfaces de las actividades.

Goedertier, De Weerdt et al [27] hacen un estudio de los principales algoritmos para descubrir BP utilizando registros de eventos para representar el comportamiento. Los registros de eventos son catálogos que almacenan ejecuciones históricas de los BP almacenados en motores de ejecución. Según el trabajo de Goedertier, De Weerdt et al, los principales algoritmos en esta área son:

- Algoritmo alfa (α) [29] y sus variaciones (α^{+}, α^{++}) [84]: permite realizar un orden binario local de las relaciones detectadas dentro de un registro de eventos. Sin embargo, este algoritmo es incapaz de descubrir ciclos cortos no locales y actividades duplicadas; y tampoco tiene en cuenta la frecuencia de las secuencias binaras que ocurren en los registros de eventos.

- Algoritmo HeuristicsMiner [28]: este método representa visualmente una matriz causal en la forma de redes heurísticas, para lo cual calcula tablas de dependencia/frecuencia desde un registro de eventos. Este algoritmo puede descubrir ciclos cortos y dependencias no locales, pero carece de la capacidad de detectar actividades duplicadas.

- Algoritmo Genetic Miner[85]: este algoritmo genético define un espacio de búsqueda en términos de matrices de causalidad que expresan dependencias de tareas. Las matrices de causalidad están estrechamente relacionadas con las redes de Petri y proveen una población inicial a partir de la cual el algoritmo detecta patrones no locales en el registro de eventos.

- Algoritmo AGNEs [86]: es un algoritmo configurable que plantea el descubrimiento de BP como un problema de clasificación multi-relacional

sobre registros de eventos. Tal como el algoritmo genético este algoritmo es capaz de construir modelos de redes de Petri desde los registros de evento analizando las restricciones frecuentes que se encuentran en estos registros. Posteriormente el algoritmo utiliza dichas restricciones para ganar entendimiento en dependencia local, dependencia no local y relaciones de paralelismo existentes entre pares de actividades. Desafortunadamente los registros de eventos raramente contienen información acerca de transiciones no permitidas y por esta razón, el algoritmo AGNEs genera dichos registros artificialmente de manera muy similar a como se hace en la actividad de aprendizaje para el contexto de minería de datos.

Como se puede observar en los trabajos de este nivel no se tiene en cuenta la semántica atribuida a las tareas internas del BP y no se aplica semántica para detectar las diferentes estructuras del control de flujo.

En general, la mayoría de los métodos analizados en esta sección aplican los niveles de descubrimiento de manera separada; sin embargo para obtener resultados con alto grado de reusabilidad y adaptación a los requisitos de usuario es necesario combinarlos [30-32] y utilizar técnicas de indexación para acelerar el proceso de descubrimiento. El presente trabajo, a diferencia de las aproximaciones presentadas en esta sección, integra técnicas de descubrimiento de los cuatro niveles descritos. En primer lugar propone un método de pre-correspondencia con alto rendimiento el cual permite recuperar BP utilizando un sistema de indexación apoyado en la semántica de comportamiento, es decir la detección de patrones de control de flujo y sus relaciones semánticas; y en segundo lugar un mecanismo de refinamiento de correspondencia aproximada basado en la comparación estructural de BP, el cual utiliza un algoritmo de isomorfismo de grafos e internamente ejecuta correspondencia lingüística (semántica y léxica) entre cada par de tareas de los BP comparados.

2.3 Resumen

Este capítulo presentó una descripción del contexto tecnológico en el cual se desenvuelve el trabajo de grado; describe las principales tecnologías como BP, WS, Web semántica; y los lenguajes y formalismos de modelado de BP. Además, en este capítulo se hizo una revisión del estado actual de conocimiento en técnicas de descubrimiento de BP y WS la cuales se clasificaron en cuatro niveles: el nivel de interfaces, centrado en los nombres, entradas y salidas de las tareas del BP; el nivel estructural, el cual evalúa isomorfamente la diferencia de dos BP representados como grafos; el nivel de semántica fundamentado en las relaciones ontológicas de las interfaces de las tareas de los BP; y el nivel de comportamiento, basado en la evaluación de aspectos como el control de flujo, registros de eventos y secuencias de mensajes.

Capítulo 3

3 Pre-correspondencia de Procesos de Negocio

En el presente trabajo de grado se presenta un entorno denominado "*BeMantics*" (Behavioral seMantics) el cual permite almacenar, comparar y recuperar BP utilizando técnicas basadas en características estructurales, lingüísticas (léxicas y semánticas) y de semántica del comportamiento.

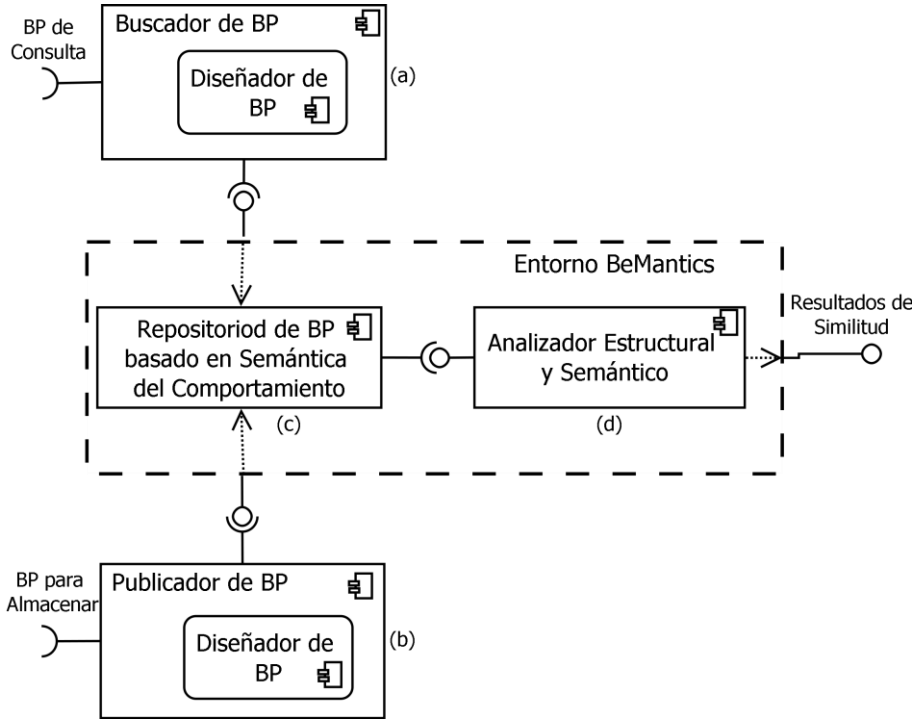


Figura 3-1: Arquitectura de referencia para el entorno de recuperación de BP.

La Figura 3-1 muestra la arquitectura de referencia de BeMantics, la cual está compuesta por dos módulos principales. El primero llamado repositorio de BP basado en semántica del comportamiento (en adelante *"repositorio"*) (Figura 3-1 (c)), el cual es responsable de almacenar, indexar y clasificar BP de acuerdo a características de semántica del comportamiento. El segundo denominado analizador estructural y semántico (Figura 3-1 (d)), el cual permite refinar los resultados del repositorio con el fin de encontrar similitudes en cuanto a semántica y estructura entre los BP almacenados en el repositorio (en adelante *"BP del repositorio"*) y un BP de consulta (en adelante *"BP consulta"*). Los otros dos módulos presentados en la Figura 3-1 son externos al entorno BeMantics y corresponden al *"Publicador de BP"* (Figura 3-1 (b)) y al *"Buscador de BP"* (Figura 3-1 (a)), los cuales permiten el diseño y enriquecimiento semántico de los BP repositorio y de los BP consulta respectivamente. En el Anexo C se explica cómo se realiza el enriquecimiento de procesos BPMO.

A continuación, en este capítulo se explica el funcionamiento del repositorio y de los módulos de diseño de BP. El cuarto módulo, el analizador estructural y semántico (Figura 3-1 (d)) se describirá con detalle en el capítulo 4 relacionado con la Correspondencia.

3.1 Grafos de Proceso

En la sección 2.1.7 se presentaron los principales formalismos de modelado de BP y se eligieron los grafos de proceso como la abstracción más sencilla y completa para describir las características de comportamiento, interfaces y estructura de los BP. En la presente sección se describe cómo se utilizaron los grafos de proceso en el entorno BeMantics a través de técnicas de isomorfismo e indexación y en el Anexo D se presenta un resumen de la teoría de grafos en general.

3.1.1 Isomorfismo de Grafos

Para comparar estructuralmente dos BP, el entorno BeMantics utilizó la técnica matemática conocida como Isomorfismo de grafos. Esta técnica es una relación de equivalencia que busca la correspondencia uno a uno entre dos grafos que preservan las relaciones de adyacencia [87]. Más formalmente un isomorfismo entre dos grafos G y G'; es un mapeo biyectivo $f: V \longrightarrow V'$ tal que $\alpha(v) = \alpha'(f(v)) \forall v \in V$. Además, para cualquier arista $e = (u, v) \in E$ existe una arista $e' = (f(u), f(v)) \in E'$ tal que $\beta(e) = \beta'(e')$ y para cualquier arista $e' = (u', v') \in E'$ existe una arista $e = (f^{-1}(u'), f^{-1}(v')) \in E$ tal que $\beta(e) = \beta'(e')$.

3.1.2 Isomorfismo de sub-grafos

El isomorfismo de grafos es útil en casos en los que no solo se requiere determinar si dos grafos son iguales, sino que también se requiere tener en cuenta si uno de ellos está contenido en otro grafo (es decir un sub-grafo). Un ejemplo claro de esto es cuando en el entorno BeMantics se quiere detectar si un grafo tiene una sub-estructura específica de comportamiento. Formalmente; si $f: V \to V'$ es un isomorfismo entre dos grafos G y G'; y G' un subgrafo de otro grafo G'', es decir $G' \subset G''$. Entonces se dice que f es un isomorfismo de sub-grafo de G'en G''. Por ejemplo, la Figura 3-2 se puede observar un grafo G y un sub-grafo G_1 de G.

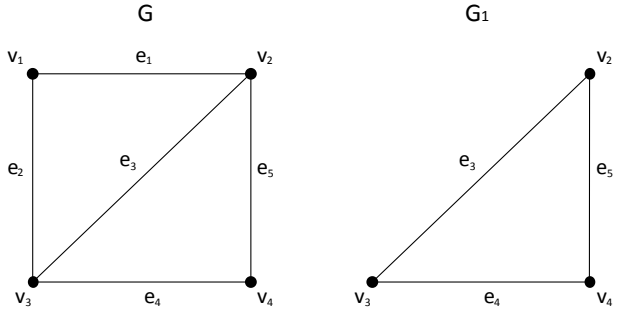


Figura 3-2: Ejemplo de un sub-grafo

3.1.3 Indexación de Grafos

La indexación de BP define índices para identificar los modelos almacenados en un repositorio de manera tal que se facilite realizar búsquedas rápidas de los

procesos o de colecciones de los mismos. En la actualidad existen diferentes clases de índices para clasificar BP los cuales generalmente están centrados en características de los procesos, como por ejemplo sus funciones de negocio (ventas, adquisiciones, finanzas, entre otros), sus características funcionales (número de tareas, número de conexiones entre la tareas) o su comportamiento (cantidad de patrones de control, cantidad de mensajes intercambiados por las tareas, entre otros). A continuación, para una mejor comprensión sobre la indexación de BP, se presentan algunos trabajos orientados a la perspectiva de la búsqueda, recuperación y minería de grafos[6].

Los trabajos de Yan y Han [88, 89] se enfocan en la recuperación de grafos basada en minería de patrones de grafos frecuentes, en la cual un grafo se considera frecuente si su frecuencia de ocurrencia en un conjunto de datos dado no es menor que un umbral mínimo. Para esto cual utilizan algoritmos de búsqueda en profundidad (DFS - Depth First Search) y descomposición de subestructuras de patrones. Otro trabajo que se enfoca en los patrones de grafos frecuentes es el trabajo de Zhu, Han et al [90], el cual describe las técnicas de minería de patrones de grafos utilizando un algoritmo de poda denominado *gprune,* el cual permite reducir el espacio de búsqueda de acuerdo a restricciones estructurales y de esta manera reforzar el proceso de minería.

Los trabajos de Giugno y Shasha [91, 92]; Ferro, Giungno et al [93]; Srinivasa [94] y Yan, Yu et al [95] definen métodos de indexación en los cuales a partir de una base de datos de grafos se encuentran todos los grafos que contienen o están contenidos en un grafo de consulta. De esta manera se puede hacer una búsqueda exacta e inexacta de patrones predefinidos dentro del grafo de consulta a través de la búsqueda de subestructuras de grafos.

El trabajo de Jin [96] integra características de varios repositorios de grafos (como los analizados en el Anexo E) con algunas de las técnicas de búsqueda y minería de grafos descritos anteriormente. Jin propone un método eficiente para la consulta de modelos de BP en repositorios, el cual toma un proceso de consulta (o un fragmento de proceso) y encuentra un conjunto de procesos almacenados en el

[6] En general la minería de grafos se puede describir como la extracción no trivial de información que reside de manera implícita en los datos de los grafos. En otras palabras prepara, sondea y explora los datos para sacar información oculta de ellos.

repositorio que contienen dicho fragmento. Los resultados se filtran a través del uso de índices, obteniendo un conjunto de procesos candidatos. Luego, se aplica una detección de isomorfismo de grafos sobre este conjunto de procesos candidatos, a través de una adaptación del algoritmo de Ullman [97].

En la presente propuesta, se estudiaron los anteriores trabajos relacionados y se seleccionó el algoritmo GraphBlast [93] como método de indexación de grafos el cual permitió realizar una búsqueda exacta e inexacta de patrones, representados como grafos, dentro de una base de datos de los grafos los BP del repositorio (BP repositorio). Para esto, GraphBlast utiliza un algoritmo de isomorfismo de grafos llamado VF2[98], el cual ofrece un mayor rendimiento en cuanto a tiempo de respuesta comparado con el algoritmo de Ullman[97].

3.2 Repositorio de Procesos de Negocio Basado en Semántica del Comportamiento

El repositorio corresponde al módulo de pre-correspondencia el cual permite almacenar, indexar y recuperar BP utilizando técnicas de la semántica del comportamiento. En este contexto la semántica del comportamiento se refiere a un mecanismo de indexación el cual direcciona el comportamiento a través de la detección de patrones de control de flujo en los BP y la semántica al tener en cuenta, no solo el número de patrones detectados, sino también, las relaciones semánticas entre los patrones (es decir las relaciones identificadas en una ontología de patrones). Para esto, el repositorio ofrece dos fases de funcionamiento (Figura 3-3). La primera denominada fase de almacenamiento en la cual se guarda un BP en el repositorio; y la segunda denominada fase de recuperación en la que se obtiene una lista ordenada de BP de acuerdo a los patrones de control de flujo detectados en un BP de consulta.

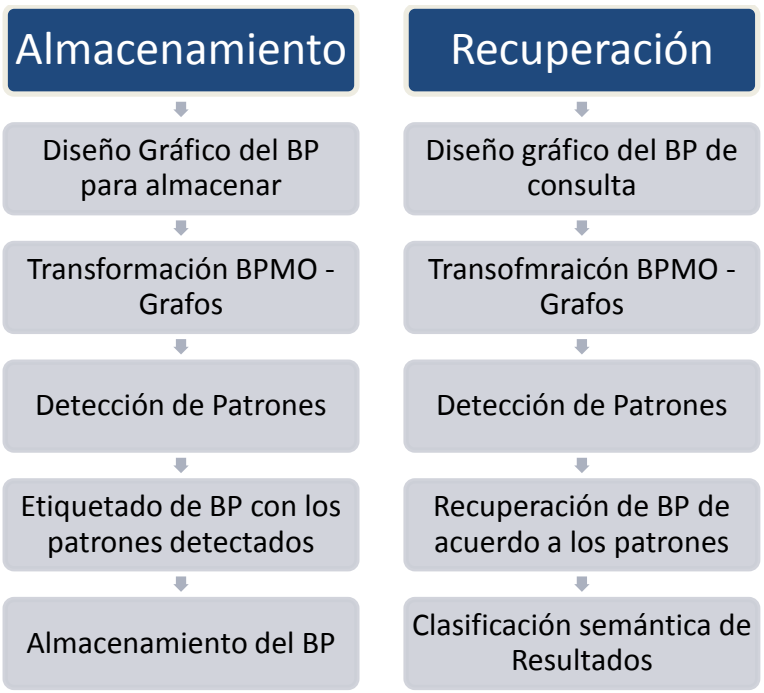


Figura 3-3: Fases de funcionamiento del repositorio

La fase de almacenamiento de BP inicia cuando un usuario modela gráficamente un proceso a través del modelador gráfico WSMO Studio 0.73 (Figura 3-1 (b)) el cual genera un modelo del proceso en el lenguaje BPMO a partir de una la notación gráfica creada por el usuario. A continuación, el proceso BPMO se transforma al formalismo de los grafos en el cual se detectan patrones de control de flujo, se adicionan etiquetas por cada patrón detectado y se almacena en el repositorio. De manera similar, en la fase de recuperación el usuario diseña gráficamente un BP de consulta el cual se transforma a un grafo de proceso. Luego, se detecta su conjunto de patrones de control de flujo y finalmente se obtiene una lista ordenada de procesos almacenados en el repositorio con un conjunto similar de patrones de control de flujo.

Teniendo en cuenta las funcionalidades descritas en los párrafos anteriores, se diseñó un modelo en capas para el repositorio, el cual se puede observar en la Figura 3-4.

Figura 3-4: capas del repositorio de BP basado en semántica del comportamiento

3.2.1 Capa de Transformación de BP

En el repositorio diseñado como producto del presente trabajo los BP transforman a dos modelos de grafos diferentes antes de ser almacenados. El primero modelo se denomina grafo de proceso y es útil en el análisis estructural de los BP; y el segundo denominado grafo TD (Trace Detection) es necesario para la detección de patrones de control de flujo y almacenamiento de los BP dentro de una base de datos de grafos llamada Berkeley DB [99], la cual está incluida como parte del repositorio. En la Figura 3-5 se puede observar el procedimiento de transformación de los BP, a partir de un archivo WSML que contiene un proceso BPMO, a un modelo de grafos de proceso y finalmente a un modelo de grafos TD.

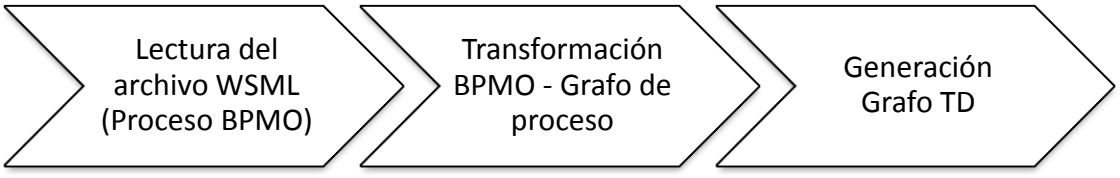


Figura 3-5 Transformación de un proceso BPMO a los modelos de grafos.

El procedimiento de transformación inicia con la lectura de un archivo WSML, el cual contiene al proceso BPMO modelado gráficamente por el usuario a través del módulo “Publicador de BP” (Figura 3-1 (b)) o del módulo “Buscador de BP” (Figura 3-1 (a)).

Este archivo WSML se recibe en un modelo de objetos java utilizando la librería WSMO4J [100] y a continuación se transforma a un grafo de proceso (Algoritmo 3-1) compuesto por nodos de tipo tarea, que representan eventos y funciones; nodos tipo conector, que representan controles de flujo (AND(split, join); OR (Split, join); XOR (split, join)); y aristas que los unen [101]. Finalmente, se genera un grafo TD el cual corresponde al grafo de proceso especificado en términos de los múltiples caminos (trazas) evaluados en cada conector (Algoritmo 3-3).

A continuación se describen los algoritmos utilizados el procedimiento de transformación de grafos:

Algoritmo 3-1: Algoritmo para la Función Convertir BPMO a Grafo de proceso

```
1   Procedure TransformBPMO2ProcessGraph (BPMO Document)
2   INPUT: BPMO_Document
3   OUTPUT: BPMOGraph
4   ADD Start node to BPMOgraph
5   ADD End nodes to BPMOgraph
6   for each wfe do
7         Calculate graph_node = BP transform(wfe)
8         ADD graph_node to BPMOGraph
9   end for
10  for each wfe_link do
11        ADD graph_edge to BPMOGraph
12  end for
    return BPMOgraph
```

El Algoritmo 3-1 toma como entrada un documento wsml que contiene el proceso BPMO (BPMO_Document) y entrega como salida un grafo de proceso (BPMOGraph). Inicialmente se fijan los nodos de inicio y fin (Líneas 3 - 4), luego se obtienen todos los elementos del proceso BPMO (wfe – workflow element), se procede a transformarlos a su contraparte en grafos (graph_node) (Líneas 5 - 8) utilizando la función BP transform (Algoritmo 3-2) y se adicionan al BPMOgraph (línea 7). A continuación se obtienen todos los enlaces que se encuentren en el proceso BPMO (wfe_link) y se transforman a aristas del grafo de proceso (graph_edge) y se adicionan al grafo de proceso resultante (BPMOGraph) (Líneas 9 – 11). La función BP transform está implementada por el Algoritmo 3-2 el cual detecta el tipo de cada elemento (wfe) del documento BPMO y a partir de éste crea

un tipo específico de nodo (conector, función o evento) el cual posteriormente se agrega al grafo de proceso en el algoritmo Algoritmo 3-1.

Algoritmo 3-2: Algoritmo para la Función BP transform

```
1   Procedure BPTransform(wfe)
2   INPUTS: wfe
3   OUTPUT: graph_node
4   if wfe ∈  DeferredChoice then
5         SET graph_node type Connector.XOR_Split
6   else if wfe ∈  Discriminator then
7         SET graph_node type Connector.XOR_Join
8   else if wfe ∈  ExclusiveChoice then
9         SET graph_node type Connector.XOR_Split
10  else if wfe ∈  InterleavedParallelRouting then
11        SET graph_node type Connector.XOR_Split
12  else if wfe ∈  MultiMerge then
13        SET graph_node type Connector.OR_Join
14  else if wfe ∈  MultipleChoice then
15        SET graph_node type Connector.OR_Split
16  else if wfe ∈  MultipleInstantiation then
17        SET graph_node type Connector.XOR_Split
18  else if wfe ∈  MultipleMergeSynchronise then
19        SET graph_node type Connector.AND_Join
20  else if wfe ∈  ParallelSplit then
21        SET graph_node type Connector.AND_Split
22  else if wfe ∈  SimpleMerge then
23        SET graph_node type Connector.XOR_Join
24  else if wfe ∈  Synchronization then
25        SET graph_node type Connector.AND_Join
26  else if wfe ∈  ErrorEvent then
27        SET graph_node type Event.ERROR
28  else if wfe ∈  ReceiveMessageEvent then
29        SET graph_node type Event.RECEIVE
30  else if wfe ∈  SendMessageEvent then
31        SET graph_node type Event.SEND
32  else if wfe ∈  TimerEvent then
33        SET graph_node type Event.TIMER
34  else if wfe ∈  Task then
35        SET graph_node type Task
36        GET taskInputs = graph_node.INPUTS
37        GET taskOutputs = graph_node.OUTPUTS
38        GET taskIdenfier = graph_node.IDENTIFIER
39  end if
40  return graph_node
```

El último algoritmo en el procedimiento de transformación es el algoritmo de detección de trazas o TDA (Algoritmo 3-3) el cual, a partir de la evaluación de cada nodo conector del grafo de proceso (BPMOGraph), crea varias rutas llamadas *"Trazas"*. Durante la creación de las trazas el algoritmo crea 4 conjuntos. El primero contiene todos los nodos visitados por el algoritmo; el segundo comprende los nodos que el algoritmo debe visitar; el tercero llamado conjunto de trazas contiene los nodos conectores que tienen la condición a ser evaluada; y el último denominado *"BackTrace"* contiene todas las trazas creadas (GrafoTD).

Algoritmo 3-3: Algoritmo TDA (detección de trazas)

```
1   procedure transformProcessGraph2GraphTD(ProcessGraph)
2   INPUT BPMOGraph G
3   OUTPUT TDGraph
4   GET StartNode i
5   ADD i to SetVisitedNodes
6   APPLY the AdjacencyFunction (i) and ADD n to SetNeighbors
7   if SetNeighbors > 1 then
8         ADD SetVisitedNodes to BackTrace
9   if SetNeighbors = 0 then end
10  while NodesToVisit > 0 then
11      GET (j) the last node ADDED to NodesToVisit
12      if (j) is the Last node of G then
13          ADD (j) to SetVisitedNodes
14          ADD SetVisitedNodes to TDGraph
15          if BackTrace > 0 then
16              GET Last Set of BackTrace
17              REPLACE SetVisitedNodes FOR BackTrace
18          return TDGraph
19      else
20          ADD (j) to SetVisitedNodes
21          APPLY the AdjacencyFunction (j) ADD m to SetNeighbors
22          if m > 1 then
23               ADD SetVisitedNodes to BackTrace
24  end while
25  return TDGraph
```

Inicialmente, el algoritmo TDA obtiene el nodo de inicio y lo adiciona en el conjunto de nodos visitados. Luego, aplica una función de adyacencia y adiciona los nodos a visitar en el conjunto de nodos vecinos (Línea 5). Si hay más de un nodo en el conjunto de nodos vecinos, entonces adiciona el conjunto de nodos visitados en el Backtrace (Línea 7). Si no hay nodos en el conjunto de nodos vecinos entonces finaliza (el primer nodo no tiene nodos vecinos) (Línea 8). Mientras existan nodos en el conjunto de nodos vecinos, el algoritmo obtiene el último nodo adicionado (j) y analiza si este es el último nodo del grafo. Si no lo es (es decir que existe una traza),

el algoritmo adiciona (j) en el conjunto de nodos visitados y coloca los nodos visitados en el grafo TD (Línea 13). Por otro lado, si el BackTrace contiene nodos, el algoritmo obtiene el último nodo y reemplaza el conjunto de nodos visitados con el último conjunto de nodos del BackTrace (Línea 16). Si el BackTrace está vacío se retorna el conjunto de trazas (el algoritmo retorna el resultado: trazas de grafos). Si (j) no es el último nodo del grafo, el algoritmo adiciona (j) en el conjunto de nodos visitados, aplica la función de adyacencia sobre (j) y adiciona los nodos vecinos (m) en el conjunto de nodos vecinos (Línea 20). Si (m) es mayor que uno, el algoritmo adiciona el conjunto de nodos visitados en el BackTrace (Línea 23).La Tabla 3-1 presenta algunos tipos de conectores definidos por el lenguaje BPMO en su versión 1.4, así como su representación en los grafos de proceso obtenidos en el Algoritmo 3-1

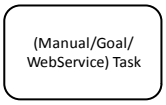
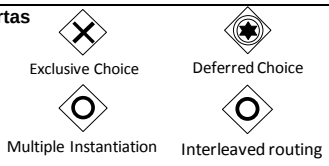
Elementos BPMO	**Etiquetas de Grafos**	**Representaciones en los grafos**
Eventos  	**Events** Los eventos se toman como nodos de tipo Evento representados por la etiqueta E	 (Start, End, Timer, Error, SendMessage, ReceiveMessage) Events
Funciones  	**Funciones** Las funciones son nodos de tipo Function representados por la etiqueta (F)	 (Manual, Goal, WebService) Tasks
Compuertas  	**Compuertas** Las compuertas son representadas por nodos Conector y pueden ser del tipo ANDS, ANDJ, ORS, ORJ, XORS XORJ)	 (DeferredChoice, ExclusiveChoice, InterleavedParallelRouting, Multiple Instantiation) Gateways
 	Los nodos del tipo XOR Split se representan con la etiqueta XORS	 (Discriminator, SimpleMerge) Gateway
 	Los nodos del tipo AND Split se representan con la etiqueta ANDS	 ParallelSplit Gateway
 	Los nodos del tipo AND Join se representan con la etiqueta ANDJ	 (MultipleMergeSynchronise, Synchronisation) Gateways
 	Los nodos del tipo OR Join se representan con la etiqueta ORJ	 Multimerge Gateway
 	Los nodos de tipo OR Split se representan con la etiqueta ORS	 MultipleChoice Gateway

Tabla 3-1: Reglas de transformación de BPMO a grafos de proceso

Para más claridad a continuación en la Figura 3-6 se presenta un ejemplo de un BP modelado como proceso BPMO y en la Figura 3-7 su representación como grafo TD.

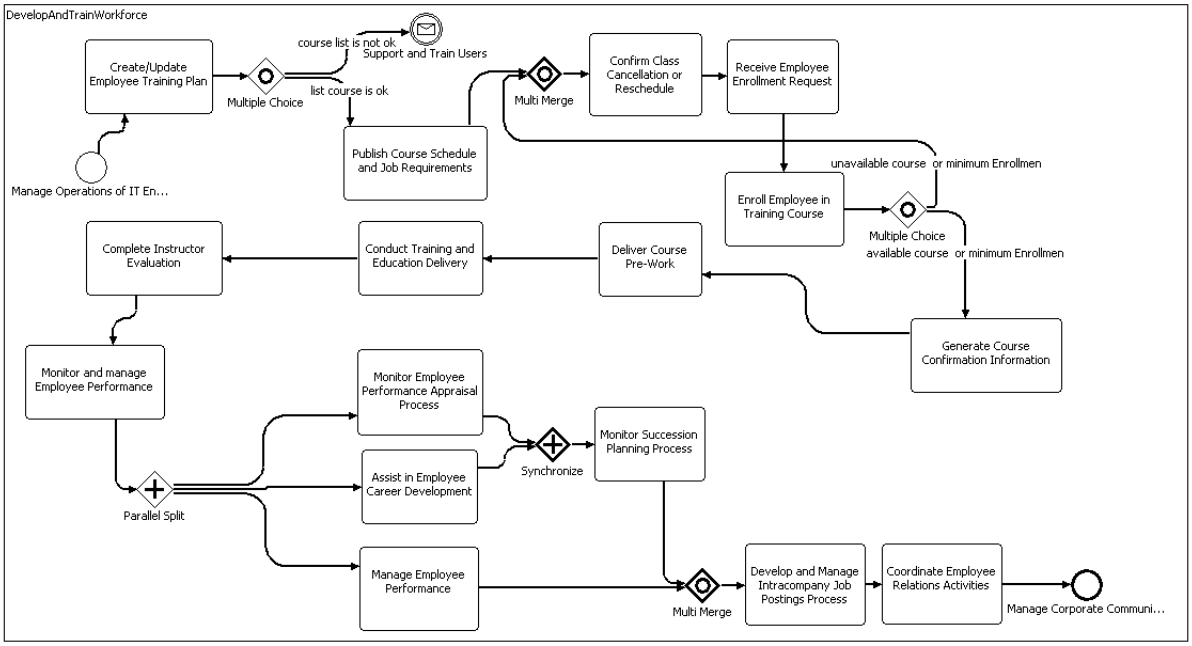


Figura 3-6: Ejemplo de un BP denominado "*Develop and Train Workforce*" descrito en WSML a partir del Modelador BPMO versión 1.4

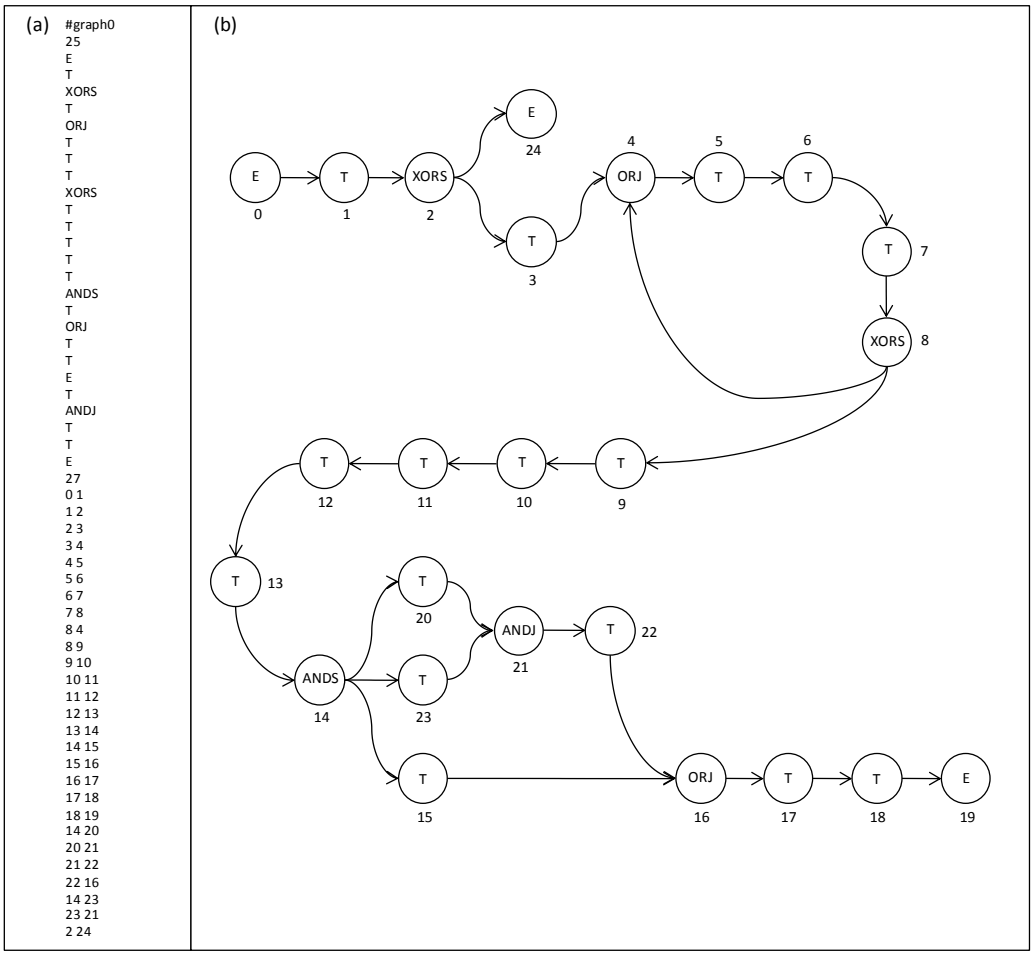


Figura 3-7: (a) Contenido del archivo .dat del Grafo BP y (b) Su representación gráfica.

La Figura 3-7 muestra el contenido del Grafo TD generado por el algoritmo TDA aplicado a un grafo de proceso obtenido a partir del proceso BPMO presentado en la Figura 3-6. En este caso un grafo TD corresponde a un documento textual almacenado con una extensión dat (.dat) y está dividido en cinco secciones (Figura 3-7 (a)). La primera sección corresponde a la primera línea del archivo, y consiste en el identificador del archivo del grafo, el cual tiene una única restricción de iniciar con el carácter “#”. La segunda sección corresponde a la segunda línea del archivo y consta del número de nodos del Grafo TD, el cual como mínimo debe ser “2”, ya que si es menor no existiría control de flujo alguno. La tercera sección hace referencia a las etiquetas de los nodos, las cuales están asociadas con la posición de los nodos en los grafos y la dirección del control de flujo según el orden en el cual aparecen al interior del archivo. Por ejemplo, la primera etiqueta “E” corresponde al nodo de

inicio, el cual tiene “0” como identificador numérico de control de flujo; la segunda etiqueta “T” corresponde al nodo de destino y tiene “1” como identificador numérico de control de flujo y así sucesivamente hasta llegar a la última etiqueta dentro de esta sección. La cuarta sección corresponde al número de aristas del Grafo TD, el cual como mínimo debe ser “1”, de acuerdo con la restricción del número de nodos de la segunda sección del Grafo TD. La quinta sección corresponde a las etiquetas de las aristas las cuales constan del identificador numérico del nodo origen y el nodo destino separados por un espacio en blanco.

En cuanto a la representación gráfica del Grafo TD (Figura 3-7 (b)), se evidencia en el comportamiento dirigido de sus aristas, la manera en la cual el algoritmo TDA asigna los identificadores numéricos de cada nodo a medida que recorre cada una de las rutas del grafo de proceso (BPMOGraph), además del etiquetado correspondiente para los elementos de control de flujo del Modelo BP de ejemplo “*Develop and train workforce.wsml*” presentado en la Figura 3-6.

3.2.2 Capa de Análisis de Patrones

Esta capa está encargada de realizar la detección de patrones de control de flujo en los grafos TD correspondientes al BP consulta y a cada BP repositorio. Como se puede observar en la Figura 3-4, está compuesta por dos sub-capas, la primera denominada detector de patrones y la segunda organizador semántico de resultados, las cuales le permiten actuar en las dos fases descritas al inicio de este capítulo. En la fase de almacenamiento, encuentra un conjunto de patrones en un grafo de proceso (BP repositorio), el cual posteriormente se etiqueta con el identificador de cada patrón detectado para facilitar su indexación y almacenamiento; y en la fase de recuperación, detecta los patrones en un BP consulta con el propósito de encontrar y clasificar por relevancia aquellos BP repositorio que contengan un conjunto de patrones similar al BP consulta. A continuación se describen en detalle las dos sub-capas.

Detector de Patrones: esta sub-capa recibe como entrada un grafo de proceso y retorna un conjunto de patrones detectados, los cuales se representan como estructuras denominadas sub-grafo. Por lo tanto, el problema de detección de

patrones se reduce a un problema de isomorfismo de sub-grafos (sección 3.1.2) el cual busca encontrar sub-grafos dentro de un grafo de proceso [92, 95, 102, 103].

En el presente trabajo se utilizó un método de isomorfismo de sub-grafos denominado GraphBlast, [92], el cual utiliza un algoritmo denominado VF2 [98] para indexar sub-estructuras contenidas en un gran conjunto de grafos (una base de datos de grafos). El algoritmo VF2 (Algoritmo 3-4) es un algoritmo de isomorfismo de grafos basado en un método de correspondencia determinístico que permite verificar correspondencias entre grafos. A través de este algoritmo es posible realizar las comparaciones entre diferentes grafos y determinar si un grafo está contenido dentro de otro (es decir que es un sub-grafo) o si son iguales en cuanto a su estructura de nodos y aristas (isomorfismo). El algoritmo es válido mientras no existan restricciones impuestas a la topología de los grafos. A continuación se presenta el algoritmo VF2 y se describe su funcionamiento.

Algoritmo 3-4: Algoritmo VF2 (isomorfismo de sub-grafos)

```
1   Procedure match (s)
2   INPUT: an intermediate state s; the initial state s_0 has M(s_0) = ∅
3   OUTPUT: The mappings between the two graphs
4         if M(s) covers all the nodes of G_2 then
5             return M(s)
6         else
7             Compute the set P(s) of the pairs candidate for
8         inclusion in M(s)
9             foreach p in P(s)
10                        if the feasibility rules succeed for the
11            inclusion of p in M(s) then
12                            Compute the state s' obtained by
13            adding p to M(s)
14                            CALL Match(s')
15                        end if
16            end foreach
17            Restore data structures
18        end if
19  end procure match
20
21  match = correspondencia
```

El algoritmo VF2 realiza un proceso de correspondencia entre dos grafos $G_1 = (N_1, B_1)$ y $G_2 = (N_2, B_2)$ a través de la determinación de un mapeo M en el cual

se asocian los nodos de G_1 con los nodos de G_2 y viceversa, de acuerdo a algunas restricciones predefinidas. Generalmente, el mapeo M se expresa como el conjunto de pares (n,m) (con $n \in G_1$ y $m \in G_2$) cada uno representando el mapeo de un nodo n de G_1 con un nodo m de G_2. De esta manera, se puede afirmar que el mapeo $M \subseteq N_1 \times N_2$ es isomorfo si y solo sí M es una función biyectiva que preserva la estructura de ramas de dos grafos y que es un isomorfismo de sub-grafo si y solo sí M es un isomorfismo entre G_2 y un subgrafo de G_1.

El proceso para encontrar la función de mapeo se describe por medio de una representación estado-espacial (SSR)[7], en la cual cada estado s del proceso de correspondencia puede ser asociado a una solución de un mapeo parcial $M(s)$ que contiene solo un subconjunto de M. El mapeo parcial $M(s)$ puede identificar unívocamente dos sub-grafos de G_1 y G_2, denominados como $G_1(s)$ y $G_2(s)$, los cuales se obtienen a través de la selección de los nodos de G_1 y G_2 incluidos en $M(s)$, y las ramas que los conectan. De acuerdo a esto, una transición de un estado genérico s a un sucesor s' representa la adición de grafos parciales asociados a s en el SSR, de un par de (n,m) nodos concurrentes. Por otra parte, entre todos los posibles estados SSR, solamente un pequeño subconjunto es consistente con el tipo de estructura requerida, en el sentido que no hay condiciones que limiten la posibilidad de alcanzar una solución completa. La consistencia de la condición se puede verificar, en caso de isomorfismo de sub-grafos, cuando los grafos parciales $G_1(s)$ y $G_2(s)$ asociados a $M(s)$ son isomorfos.

En la presente propuesta se utilizó GraphBlast con el fin de encontrar los patrones de control de flujo representados por sub-estructuras dentro de los grafos TD de cada uno de los BP repositorio. Cada grafo TD, corresponde a una Lista de Nodos y Aristas (LNE por sus siglas en inglés) [102] basada en nodos y caminos. Los nodos se etiquetan con un número (id-nodo – *id-node*) y una etiqueta (etiqueta-nodo – label-node); y los caminos se etiquetan con una lista de números id-nodo (conocida como id-camino – id-path) y una lista de etiquetas etiqueta-nodo (conocida como etiqueta-camino – label-path) con aristas sin etiquetas entre cada par de nodos consecutivos (Figura 3-7). Por lo tanto se construye un índice buscando todos los caminos que

[7] SSR (State Space Representation) es un modelo matemático que representa a un sistema físico como un conjunto de entradas, salidas y variables de estado relacionadas por ecuaciones diferencias de primer orden.

tienen una longitud predeterminada y comienzan en un nodo determinado. Por ejemplo, un *etiqueta-camino* es: EANDS, y un id-camino sería: (0,1).

En este caso para cada grafo y para cada nodo, se encuentran todos los trayectos que empiezan en un nodo específico y tienen una longitud desde uno hasta un tamaño predefinido utilizando una variable denominada longitud de camino (LP), la cual se fija por defecto al valor de 4 (*LP=4*). La construcción del índice se realiza a través de un conjunto de listas *id-camino* e *id-etiqueta* utilizando una tabla de llaves (*hash table*), cuyas llaves son los valores de la lista *etiqueta-camino*. En el presente proyecto se utiliza un lista *etiqueta-camino* para definir y describir un grafo de un patrón que se busca dentro de un grafo TD; por ejemplo, una *etiqueta-camino* $h(TTXORJT)$ describe un patrón de grafo predefinido el cual contiene dos tareas (T) iniciales, un conector XOR-Join (XORJ) y una tarea final. Por lo tanto, la construcción de los índices se hace por el número de ocurrencias de esta *etiqueta-camino* dentro de cada grafo almacenado Figura 3-8 (b).

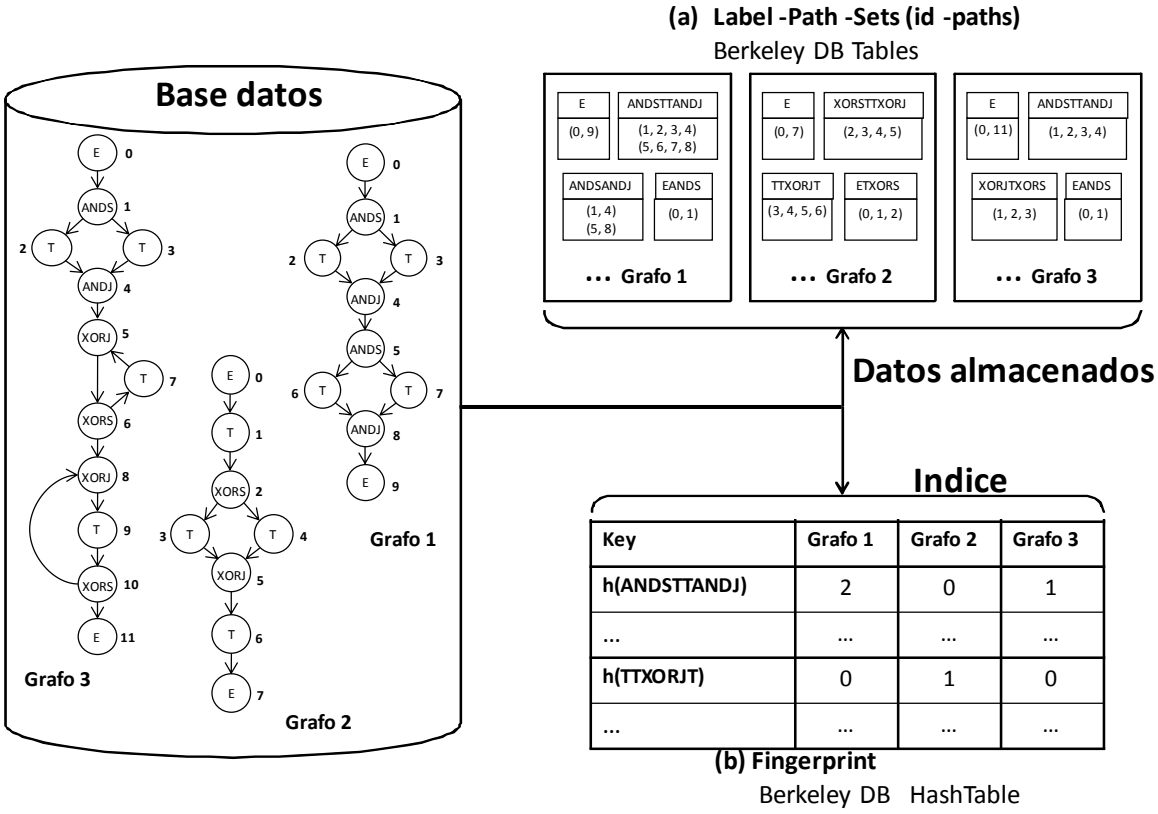


Figura 3-8: Almacenamiento de los grafos TD y la creación del índice sobre las tablas de la Base de datos de Berkeley

La tabla de llaves se define como una huella digital de la base de datos y está compuesta por una matriz donde las filas están etiquetadas como una lista *etiqueta-camino* y cada columna se asocia a un grafo almacenado en la base de datos. Toda la información relacionada a la construcción del índice se almacena en la base de datos Berkeley [104]. La Figura 3-8(a) muestra una lista de *id-camino* de todos los trayectos que representan una secuencia de etiquetas en una lista etiqueta-camino.

Organizador Semántico de Resultados: Esta sub-capa organiza los resultados obtenidos en la sub-capa de recuperación de BP repositorio de acuerdo a cinco distancias representadas como diferencias numéricas respecto a un grafo de consulta:

- **Distancia por número de patrones (*Dp*):** evalúa la distancia en términos de la diferencia entre el conjunto de patrones de BP repositorio ($\boldsymbol{P_T}$) y el conjunto de patrones de un BP consulta ($\boldsymbol{P_Q}$).

$$Dp = \frac{\left|P_Q - P_T\right|}{P_Q + P_T}$$

Ecuación 3-1

- **Distancia por número de nodos (*Dn*):** evalúa la distancia de acuerdo al número de nodos tipo conector (ANDS, ANDJ, XORJ, XORS, ORS, ORJ) que se encuentran en un BP repositorio ($\mathrm{N_t}$) y los que se encuentran en un BP consulta ($\mathrm{N_q}$).

$$Dn = \frac{\left|N_q - N_t\right|}{N_q + N_t}$$

Ecuación 3-2

- **Distancia por número de aristas (*De*):** evalúa el número de aristas que se encuentran en un BP repositorio (*Et*) y el número de aristas en un BP de consulta (*Eq*).

$$De = \frac{|E_q - E_t|}{E_q + E_t}$$

Ecuación 3-3

- **Distancia de descriptor de ruta (Drd_T):** evalúa la distancia en términos del número de ocurrencias y posiciones de los patrones detectados en BP repositorio (oNt y PPt) y los detectados en un BP consulta (oNq y PPq). En este sentido la distancia de descriptor de ruta corresponde a la suma de todas las distancias de descriptor de ruta de cada patrón detectado en los dos BP.

$$Drd_T = \sum \frac{\frac{|PP_q - PP_t|}{PP_q + PP_t}}{|oN_q - oN_t|}$$

Ecuación 3-4

Por ejemplo, supóngase que se tienen dos BP: BP_1 y BP_2, los cuales comparten los patrones XORS y XORJ en común, entonces se construye una matriz con las ocurrencias y posiciones de los patrones en los BP como se puede observar en la Figura 3-9.

BP_1	**BP_2**
(1) XORJ/6	(1) XORS/2
(2) XORS/9	(2) XORJ/4
	(3) XORS/7
	(4) XORS/10

Ocurrencias de Patrones

Posición del patrón

Figura 3-9: ejemplo de ocurrencias y posiciones de los patrones en dos BP

Entonces, de acuerdo a la Figura 3-9, la distancia Drd para el primer patrón (XORJ) se puede calcular como:

$$Drd_1 = \frac{\frac{|\mathrm{PP_q} - \mathrm{PP_t}|}{\mathrm{PP_q} + \mathrm{PP_t}}}{|\mathrm{oN_q} - \mathrm{oN_t}|} = \frac{\frac{|6-4|}{6+4}}{|1-2|} = \frac{1}{5}$$

Y para el Segundo patrón (ORS) como:

$$Drd_2 = \frac{\frac{|\mathrm{PP_q} - \mathrm{PP_t}|}{\mathrm{PP_q} + \mathrm{PP_t}}}{|\mathrm{oN_q} - \mathrm{oN_t}|} = \frac{\frac{|9-2|}{9+2}}{|2-1|} = \frac{7}{11}.$$

Por lo tanto, la distancia total de descriptor de ruta es:

$$Drd_T = \frac{\mathrm{DDR_1} + \mathrm{DDR_2}}{\text{number of DDRs}} = \frac{\frac{1}{5} + \frac{7}{11}}{2} = 0{,}41818.$$

- **Distancia de patrones semánticas (*Dsp*):** esta distancia evalúa la diferencia entre dos patrones de acuerdo a relaciones generales entre ellos establecidas por Rusell et al [54]. Para esto en la presente propuesta se creó una abstracción de relaciones entre 12 patrones denominada *ontología de patrones de control de flujo* (Anexo A.7), la cual tiene dos relaciones principales entre los patrones: especialización, cuando un patrón es una vista más restringida de otro; y composición cuando un patrón se puede representar como la unión de otros patrones. De esta manera se utilizó la ontología de patrones con el fin de encontrar una distancia semántica entre patrones contenidos en BP. La distancia semántica se obtiene calculando una distancia de salto $(\mathrm{D_{salto}})$ [105] entre dos patrones en la ontología.

$$\mathrm{D_{salto}} = 1 + \frac{1}{2^{\text{profundidad}}}$$

Ecuación 3-5

En la Ecuación 3-5 el término "*profundidad*" se refiere al número de saltos en la ontología desde la raíz hasta el concepto objetivo. De esta manera la distancia semántica total es la suma de todos los valores sobre cada camino entre un par de conceptos:

$$Dsp = \sum \mathrm{D_{salto}}$$

Ecuación 3-6

A continuación se describe el algoritmo para calcular esta distancia semántica (Algoritmo 3-5). El algoritmo empieza por evaluar si el concepto del enriquecimiento de un patrón de comportamiento (C_Q) es diferente al concepto

de otro patrón de la ontología (C_T), ya que si esta condición no se cumple significa que los dos enriquecimientos corresponden exactamente al mismo concepto, por lo cual la distancia entre ellos toma el valor de 0.

En el caso en que C_Q y C_T sean distintos se procede a encontrar los superconceptos (denotados por SC_Q y SC_T respectivamente) y el concepto que es común en estos dos arreglos (CC) con el fin de definir el trayecto más corto entre ellos (menor número de saltos). Posteriormente se calcula la sumatoria de todas las distancias por salto que hay entre CC y C_Q (en el arreglo SC_Q) y entre CC y C_T (en el arreglo SC_T). Por último estos dos valores se suman para obtener la Distancia Semántica total (Dsp) entre los dos conceptos.

Algoritmo 3-5: Algoritmo para Función distancia semántica entre conceptos**.**

```
1   INPUT: C_Q, C_T
2   OUTPUT: Dsp
3
4   Dsp := 0
5   if C_Q ≠ C_T then
6   begin
7        saltos := 0
8        SC_Q [ ] := getSuperConcepts(C_Q)
9        SC_T [ ] := getSuperConcepts(C_T)
10       CC := findCommonConcept(SC_Q, SC_T)
11       dist1 := 0
12       dist2 := 0
13       for j := 1 to indexOf(CC, SC_Q)
14            dist1 := dist1 + 1 + 1/(2^(size(SC_Q)-j))
15       for j := 1 to indexOf(CC, SC_T)
16            dist2 := dist2 + 1 + 1/(2^( size(SC_T)-j))
17
18       Dsp := dist1 + dist2
19  end

    Dsp es la distancia semántica entre conceptos C_Q y C_T
    saltos es el número de saltos entre C_Q y C_T
    CC el el concepto común más cercano entre SC_Q y SC_T
    dist1 es el número de saltos entre C_Q y CC
    dist2 el número de satos entre C_T y CC
    indexOf(CC, SC_Q) retorna el valor de índice para CC en
    SC_Q
```

Finalmente la distancia total de patrones (DpT) entre un BP consulta y un BP repositorio se puede calcular como la suma de las cinco distancias presentadas anteriormente.

$$DpT = Dp + Dn + De + Drd_T + Dsp$$

Ecuación 3-7

De esta manera los BP repositorio se pueden clasificar en cinco listas de acuerdo a cada una de las distancias presentadas anteriormente o en una sola lista regida por la distancia total (DpT), es decir empezando por aquellos con la menor DpT hasta aquellos con la mayor DpT respecto al BP de consulta. Finalmente la lista que se utilice pasa al módulo de Correspondencia estructural y semántica en el cual se ejecuta una comparación estructural exhaustiva, uno a uno, de cada BP repositorio ordenado en la lista, con el BP consulta previamente seleccionado.

En resumen, los algoritmos Algoritmo 3-6 y Algoritmo 3-7, las fases de almacenamiento y recuperación del repositorio en los respectivamente.

Algoritmo 3-6: Algoritmo para almacenar procesos BPMO

```
1   INPUTS: BPMO document
2   pg = transformBPMO2ProcessGraph(BPMO document) (ALGORITMO 3.1)
3   tdg = transformProcessGraph2GraphTD(pg) (ALGORITMO 3.3)
4   GET patternsSet
5   for each p ∈ patternsSet
6
7       EXECUTE VF2 para buscar p en tdg (ALGORITMO 3.4)
8       if p ⊆ tdg then
9         ADD p to detectedPatterns
10      end if
11  end for
12  STORE BPMO document
13  STORE p con con detectedPatterns
14
    BPMO Document es el documento con formato WSML generado
    gráficamente por el usuario
    pg es la representación del BP como grafo de proceso
    tdg es la representación dl BP como grafo TD (con detección de
    trazas)
    patternsSet es el conjunto de 12 patrones seleccionado en este
    proyecto
    p es cada uno de los patrones del conjunto patternsSet
    detectedPatterns es el conjunto de patrones encontrados por el
    algoritmo VF2 dentro de un gtd
```

Algoritmo 3-7: Algoritmo para recuperar y clasificar procesos BPMO

```
1   INPUTS: BPMO de consulta
2   OUTPUTS: rankingList
3   pg = transformBPMO2ProcessGraph(BPMO de consulta) (ALGORITMO 3.1)
4   tdg = transformProcessGraph2GraphTD(pg) (ALGORITMO 3.3)
5   GET patternsSet
6   for each p ∈ patternsSet
7       EXECUTE VF2 para buscar p en tdg (ALGORITMO 3.4)
8       if ∃ p en gtd then
9         ADD p to detectedPatterns
10      end if
11  end for
12  if detectedPatterns no está vacío then
13      similarBPlist = findBPwithPatterns(detectedPatterns, tdg)
14  end if
15  for each BP repositorio ∈ similarBPlist
16      CALCULATE DpT = Dp + Dn + De + Drd_T + Dsp
17      ADD BP repositorio to rankingList according to DpT
18  end for
19  return rankingList
```

BPMO de consulta es el documento con formato WSML generado gráficamente por el usuario como consulta

consulta SQL para encontrar todos los BP repositorio con un conjunto *detectedPatterns* similar al del BP consulta
pg es la representación del BP como grafo de proceso
tdg es la representación dl BP como grafo TD (con detección de trazas)
patternsSet es el conjunto de 12 patrones seleccionado en este proyecto
p es cada uno de los patrones del conjunto *patternsSet*
detectedPatterns es el conjunto de patrones encontrados por el algoritmo VF2 dentro de un *tdg*
rankingList es lista ordenada de BP repositorio de acuerdo a *DpT*
Las variables DpT, Dp, Dn, De, Drd_T y Dsp se en las ecuaciones Ecuación 3-1 - Ecuación 3-7

3.2.3 Capa de Almacenamiento

Como se puede observar en la Figura 3-4, esta capa tiene tres repositorios en los cuales se almacenan los modelos de BP representados como BPMO, grafos TD y las referencias de los BP (BP repositorio y BP consulta). A continuación se describe la

funcionalidad de estos repositorios dentro de la presente propuesta y su contenido (Figura 3-10).

Modelos BP	**Grafos TD**			**Referencias BP**
Archivos WSML (Procesos BPMO)	*BP repositorio*	*BP consulta*	*Patrones de control de flujo*	*RDBMS*

Figura 3-10: Contenido de los repositorios de la capa de almacenamiento

El primer repositorio llamado "*Modelos BP"*, se encarga de almacenar los procesos BPMO en su forma original a través del formato WSML (archivos con extensión .wsdl). El segundo, denominado "*Grafos TD"*, se encarga de almacenar los grafos TD tanto de los BP repositorio, los BP consulta y de los 12 patrones de control de flujo elegidos para esta propuesta (Anexo A) en formato de trazas (archivos con extensión .dat). En este segundo repositorio todos los grafos TD de los BP consulta y los BP repositorio se almacenan en una base de datos Berkeley [99]. El último repositorio denominado *"Referencias BP"* almacena la ubicación de los documentos WSML y la referencia de éstos con el respectivo grafo TD y los patrones que contiene en una base de datos relacional. En la Figura 3-11 se puede observar la abstracción del modelo entidad relación correspondiente a la base de datos relacional del repositorio *Referencias BP.*

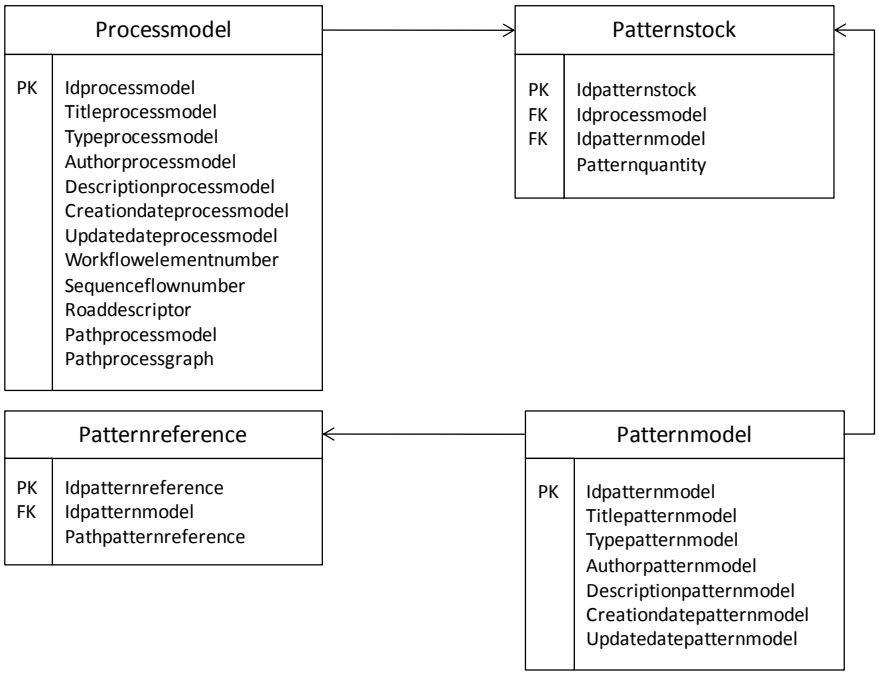


Figura 3-11: Abstracción del diagrama de entidad relación de la base de datos de referencias

3.3 Resumen

En este capítulo se presentó el módulo de pre-correspondencia constituido por un repositorio de BP basado en semántica de comportamiento, el cual permite almacenar BP teniendo en cuenta patrones de control del flujo. Además facilita la recuperación de una lista ordenada de BP de acuerdo al conjunto de patrones de control de flujo encontrados en un BP de consulta. En la generación de la lista ordenada se tienen en cuenta las relaciones semánticas que existen en los patrones de control de flujo, las cuales se establecieron a través de una ontología de patrones de control de flujo.

Capítulo 4

4 Correspondencia

Este capítulo describe el mecanismo de correspondencia estructural y semántica implementada en la presente propuesta por el módulo analizador estructural y semántico (Figura 3-1(d)), el cual ejecuta un refinamiento de los resultados obtenidos por el repositorio. Por lo tanto recibe como entradas la lista de BP entregada por el módulo de clasificación semántica y el BP consulta utilizado para generarla (ver sección 3.2). No obstante, a diferencia del repositorio que trabaja con la representación llamada *"grafo TD",* en este módulo se utiliza la representación de *"grafos de proceso"* (ver sección 3.2.1), por tanto, cuando en este capítulo se nombra el término BP repositorio se refiere a los grafos de proceso de los BP repositorio y de la misma manera con el BP consulta.

Este módulo genera correspondencias entre el BP consulta y cada uno de los BP repositorio, para lo cual ejecuta una serie de operaciones de edición que modifican a cada uno de los BP repositorio con el fin de hacerlos tan similares como sea posible al BP consulta. Cada operación de edición supone un costo de edición y por lo tanto entre más operaciones se realicen en un BP repositorio, más diferente será este respecto al BP consulta. Al resultado de estas operaciones de edición se le denomina *"grafo de resultado"*, el cual refleja los cambios aplicados al BP repositorio. Por lo tanto en cada ejecución del módulo de correspondencia se tendrá un grafo de resultado por cada BP repositorio de acuerdo a un solo BP consulta.

Finalmente, los grafos de resultado se evalúan en términos de estructura (número de operaciones de edición realizadas), lingüística (diferencias léxicas y semánticas entre los conceptos asociados a las interfaces de las tareas) y comportamiento secuencial (número de secuencias similares) con el fin de entregar como resultado una lista de correspondencias entre el BP consulta y cada uno de los grafos de resultado relacionados con los BP repositorio. La Figura 4-1 muestra los componentes de este módulo, los cuales se describen a continuación.

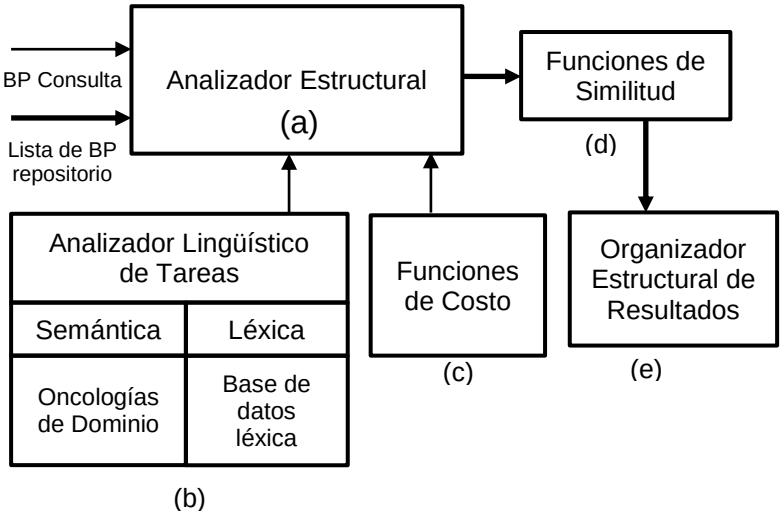


Figura 4-1: Analizador semántico y estructural

4.1 Analizador de Correspondencia Estructural

El analizador estructural (Figura 4-1 (a)) recibe como entrada un BP consulta y lo compara estructuralmente con cada uno de los BP repositorio (es decir aquellos grafos que se obtuvieron como resultado de la clasificación realizada por el repositorio utilizando semántica del comportamiento). La correspondencia estructural, ejecutada en este módulo, está basada en el algoritmo de isomorfismo de sub-grafos por corrección de errores [106] previamente implementado por Corrales et al [21], el cual empieza con la creación de un conjunto de mapeos desde un nodo de un BP consulta a cada nodo de cada BP repositorio. Cada mapeo involucra un conjunto de

operaciones de edición (operaciones de modificación en un BP repositorio como eliminar, insertar o substituir nodos y aristas), las cuales a su vez implican costos de edición. De esta manera se puede estimar un costo total para cada mapeo sumando el costo individual de cada operación. El procedimiento se repite hasta que se encuentre un mapeo mínimo (un mapeo con un costo mínimo de edición) o cuando todos los mapeos posibles han excedido un valor de tolerancia (*AC – Acceptable Cost*) predefinido por el usuario.

En la presente investigación, los grafos de proceso (BP consulta y BP repositorio) están compuestos por nodos de tipo tarea, los cuales representan actividades de negocio; conectores los cuales expresan restricciones de control de flujo; y aristas que conforman enlaces entre los nodos. Por lo tanto, se ha definido un conjunto de operaciones de edición válidas que el analizador estructural puede ejecutar; estas son: substituir o eliminar nodos de tipo tarea, insertar o eliminar conectores, e insertar y eliminar aristas; cada una de las cuales tiene un costo de edición asociado representado (distancia respecto al BP consulta). Además, se definen algunos factores de multiplicación para permitir que los usuarios puedan personalizar las restricciones del analizador estructural con el propósito de ejecutar una operación de edición con mayor probabilidad que otras. Por ejemplo, si un usuario selecciona el valor 0,2 para la operación de sustitución de nodos y el valor 0,5 para la operación de eliminación de nodos, entonces el algoritmo preferirá editar el nodo en lugar de removerlo.

En el presente trabajo el algoritmo de isomorfismo con corrección de errores que se implementó se presenta en el Algoritmo 4-1, el cual con el fin de reducir el espacio de búsqueda discrimina el tipo de nodos a comparar (función, conector, evento) y crea grupos con tipos similares de dos grafos de proceso distintos un BP consulta (G_Q) y BP repositorio (G_T). De esta manera cuando se hace correspondencia entre los dos grafos G_Q y G_T solo se realizan comparaciones entre nodos que pertenecen al mismo tipo.

El Algoritmo 4-1 empieza mapeando el primer nodo incluido en el primer grupo de G_Q con todos los nodos del mismo conjunto de G_T y elige el mejor mapeo (con un costo mínimo). Esto representa un mapeo parcial (*p'*) que posteriormente se por medio del mapeo de un vértice *v* (en el grafo objetivo que aún no ha sido mapeado) a

un vértice *w* en el grafo de consulta (que no pertenece al mapeo actual) lo cual implica la ejecución de operaciones de edición de nodos y operaciones de edición de aristas (*FunctionMatch, EventMath, EdgeCost y ConnectorCost*) en las cuales, los atributos de *v* se deben sustituir por atributos de *w*, y para cada par de nodos ya mapeados (v', w') se debe asegurar que cada arista (v', v) en el grafo consulta puedas ser mapeado a una arista (w', w) en el grafo objetivo por medio de operaciones de edición de arista [106]. El proceso termina cuando se alcanza un estado representando un isomorfismo de subgrafos con corrección de error de G_Q a G_T; o cuando todos los estados en el espacio de búsqueda tienen costos de edición que excedan el umbral de aceptación. El costo del mapeo $C(p')$ representa el costo de extender el mapeo actual *p* con el siguiente nodo en el grafo consulta.

Algoritmo 4-1: Algoritmo de Correspondencia Estructural y Semántica

```
1   INPUT: GQ(VQ), GT(VT)
2   Inicializa OPEN: mapea cada nodo VQ en cada nodo de VT
3   CREATE p
4   CALCULATE C(p)
5   ADD p a OPEN
6   if OPEN esta vació then EXIT.
7   SELECT p de OPEN tal que C(p) sea mínimo y remover p de OPEN
8   if C(p)> AC then EXIT.
9   if p representa un mapeo completo de GQ a GT then output p.
10
11  SET AC = C(p).
12  GO TO línea 5
13  sea p = {(v1 , wi),…, (vk , wj )} el mapeo actual de k nodos de GQ.
14  for each w en VT que no ha sido mapeado a un nodo en VQ
15     Extender el mapeo actual p a p′ mapeando el nodo vk+1 de V a w
16      p′ = {(v1 , wi),…, (vk , wj ), (vk+1, w)}
17      CALCULATE C(p′)
18        if (vk+1, w)ϵ Función_Set then EXECUTE FunctionMatch
19        if (vk+1, w)ϵ Evento_Set then EXECUTE EventMath
20      ADD p′ a OPEN
21      if ∃ e(w, w′) then EXECUTE EdgeCost
22
23      if ∃ connector(w, w′) EXECUTE ConnectorCost
24      else EXECUTE SupressionCost (v, v′)
25      ADD Distance(EdgeCost, ConnectorCost, SupressionCost) to C(p′)
26  GOTO línea 5

    p es un mapeo de VQ en VT
    w es un nodo de interacción
    C(p) es el costo del mapeo p
    Funcion_Set: conjunto de todos los nodos de tipo Función
```

```
Evento_Set: conjunto de todos los nodos de tipo Evento
Conector_Set: conjunto de todos los nodos de tipo Conector
e(w,w') es una arista entre los nodos w y w'
connector(w,w') es un conector entre los nodos w y w'
```

Los costos de edición se calculan con las funciones *FunctionMatch, EventMath, EdgeCost y ConnectorCost.;* y el costo total se calcula sumando los valores de las repuestas de ellas. El módulo que se encarga de ejecutar dichas funciones se denomina módulo de funciones de costo (Figura 4-1 (c)), el cual se explica en detalle a continuación.

4.2 Funciones de Costo

El módulo de funciones de costo Figura 4-1 (c) calcula los costos de las operaciones de edición para estimar la distancia entre un BP consulta y cada uno de los BP repositorio. A continuación se realiza una descripción de las fórmulas utilizadas para el cálculo de costos de edición y los algoritmos correspondientes.

- ***Distancia de arista (ed):*** esta función cuenta el número de aristas eliminadas (*de*) o insertadas (*ie*) en cada BP repositorio y multiplica este valor por el factor de arista eliminada (ϑ) y el factor de insertar aristas (μ) predefinidas por el usuario.

  En este caso para cada mapeo $M(v,w)$ y $M'(v',w')$ (dónde $(v,v') \in$ BP consulta y $(w,w') \in$ BP repositorio) y supuesto que existe una arista entre los nodos (v,v'), el
  Algoritmo 4-2, analiza si las operaciones de insertar, substituir una arista o eliminar un conector entre (w,w') son necesarias.

Algoritmo 4-2: Algoritmo para la función *EdgeCost*

```
1  INPUTS: M(v,w) y M'(v',w')
2  OUTPUT: ed
3  Supóngase los mapeos M(v,w) y M'(v',w') donde (v,v') ∈ BP consulta
4  y (w,w') ∈ BP repositorio y supuesto que existe una arista entre
5  los nodos (v,v')
6
```

```
for M(v,w) y M'(v',w')
   if ∃ e(w,w') then
      return ed = 0;
   else if ∄ e(w,w') then
      return ed = ϑ * ie
   else if ∃ connector(w,w') then
      return ed = ϑ * ie + μ * de
end if

ed es la distancia de arista
ie es el costo de insertar una arista
de es el costo de eliminar una arista
e(w,w') es una arista entre los nodos w y w'
connector(w,w') es un conector entre los nodos w y w'
```

- ***Distancia de Conector (cd):*** esta función cuenta el número de conectores substituidos (en el caso que las dos conectores sean de tipos diferentes), insertados y eliminados en cada BP repositorio y los multiplica por un factor de conector substituido (ω), conector eliminado (ρ) y un factor de conector insertado (σ) predefinido por el usuario.

  En este caso para cada mapeo $M(v,w)$ y $M'(v',w')$ (dónde $(v,v') \in$ BP consulta y $(w,w') \in$ BP repositorio) y supuesto que existe un conector entre los nodos (v,v'), el Algoritmo 4-3, analiza si las operaciones de insertar, substituir un conector o eliminar una arista entre (w,w') son necesarias.

Algoritmo 4-3: Algoritmo para la función *ConnectorCost*

```
INPUTS: M(v,w) y M'(v',w')
OUTPUT: cd
Supóngase los mapeos M(v,w) y M'(v',w') donde (v,v') ∈ BP consulta
y (w,w') ∈ BP repositorio y supuesto que existe un conector
entre los nodos (v,v')
for M(v,w) y M'(v',w')
   if ∃ connector(w,w') then
      if connector(w,w') type ≠ connector(v,v') type then
         return cd = ω * sc
      else if connector(w,w') type = connector(v,b') type then
         cd = 0
      end if
   else if ∄ connector(w,w') then
      return cd = ϑ * ic
   end if
```

```
17      if ∃ e(w,w′) then
18        return cd = ρ ∑ dc + σ ∑ ic
19
20      end if
```

cd es la distancia de conector
sc es el costo de sustituir un conector por otro de diferente tipo
ic es el costo de insertar una arista
dc es el costo de eliminar una arista
$e(w,w')$ es una arista entre los nodos w y w'
$connector(w,w')$ es un conector entre los nodos w y w'

- ***Distancia de Nodo tipo Tarea (tnd)*:** esta función cuenta el número de nodos tipo tarea eliminados (dn) y el costo de substituir un nodo del BP consulta por un nodo de un BP repositorio (sn) (es decir, encontrar un nodo de tipo tarea en el BP repositorio, el cual se pueda asignar al mapeo como nodo que reemplace a un nodo del BP consulta debido a sus similitudes en términos de nombres de las tareas e interfaces (entradas/salidas)). La operación de substitución de nodos tipo tarea se calcula utilizando el módulo analizador lingüístico (Figura 4-1(b)) el cual se describe en la sección 4.3. Tal como las otras operaciones, esta función utiliza factores de multiplicación predefinidos por los usuarios, en este caso son: el factor de nodo eliminado (τ) y el factor de nodo substituido (φ).

$$tnd = \tau \sum dn + \varphi \sum sn$$

Ecuación 4-1

- ***Distancia Total (TD):*** la distancia total suma las distancias anteriores con lo cual se puede organizar los BP repositorio en una lista (ranking) basada en la distancia entre cada uno de ellos y el BP consulta.

$$TD = ed + cd + tnd$$

Ecuación 4-2

4.3 Analizador Lingüístico

El analizador lingüístico (Figura 4-1(b)) evalúa la operación de sustitución comparando los nodos tipo tarea de un BP consulta y un BP repositorio. Los nodos tipos tarea que se comparan en esta propuesta están condicionados por el lenguaje BPMO el cual los clasifica en: nodos evento, (tareas ejecutadas de acuerdo a estímulos como por ejemplo, una alarma, un error o el fin de un periodo de tiempo); y como nodos función (actividades del BP que tienen nombre e interfaces (entradas y salidas)) con la capacidad de recibir anotaciones semánticas. Por lo tanto para evaluar la operación de sustitución para nodos función se utilizó un analizador semántico y para los nodos de tipo evento un analizador léxico, ya que estos últimos no tienen enriquecimiento semántico.

4.3.1 Analizador Léxico

Este analizador estima el costo de la operación de sustitución en términos de una distancia léxica entre dos nodos tipo tarea, los cuales no tienen enriquecimiento semántico. La distancia léxica evalúa la diferencia entre el nombre de dos nodos, analizando combinaciones de palabras y abreviaciones.

En esta propuesta, para encontrar la distancia léxica, se utilizaron existentes como el algoritmo NGram para estimar la similitud de acuerdo al número de secuencias comunes de una longitud de caracteres definida (q-gramas) entre los nombres de los nodos. El algoritmo Check abbreviation que utiliza un diccionario de abreviaciones, y el algoritmo Check synonym, el cual encuentra sinónimos utilizando una base de datos léxica (por ejemplo WordNet [34]). Adicionalmente, se utilizó la ecuación propuesta por Patil et al [107], la cual utiliza el resultado de los algoritmos NGram (*m1*), Check Synonym (*m2*) y Check abbreviation (*m3*) para evaluar la distancia léxica (LS) entre dos nombres de nodos.

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

Ecuación 4-3

4.3.2 Analizador Semántico

El analizador semántico tiene como fin encontrar un costo de sustitución basado en cálculos de similitud entre dos nodos tipo función enriquecidos semánticamente. El enriquecimiento semántico se realiza sobre los identificadores (nombres) de las tareas y sobre sus interfaces (entradas y salidas), utilizando dos ontologías de dominio. En la presente investigación se utilizaron dos ontologías del dominio de las telecomunicaciones seTOM (enhanced Telecom Operations Map) [108, 109] y sSID (Shared Information and Data model) [110], las cuales son parte del entorno de ontologías YATOSP (Yet Another Telecoms Ontology, Service and Process) [111].

La ontología seTOM se utilizó para enriquecer semánticamente los identificadores de las tareas y la ontología sSID para las interfaces. De esta manera el cálculo de la similitud semántica se basa en la distancia semántica expresada en la sección de clasificación semántica de BP (sección 3.2.2) del repositorio, para lo cual se utilizó el sofware WSML2reasoner [112] el cual facilita el cálculo de la distancia $\mathrm{D}_{\mathrm{salto}}$ entre dos conceptos. Adicionalmente, en el Anexo C se presenta un resumen acerca del enriquecimiento semántico de BP y en el Anexo B.4 se hace un estudio de los modelos eTOM y SID.

A continuación se describe el cálculo de la similitud semántica aplicada tanto a los identificadores (nombres de las tareas) como a las interfaces.

- **Similitud Semántica de Identificadores:** este análisis se realizó teniendo en cuenta los niveles 2 y 3 del modelo eTOM en los cuales se definieron 5 categorías de correspondencia entre los identificadores.

- **Exact:** Cuando los dos términos del enriquecimiento comparados pertenecen al mismo concepto en la ontología SeTOM.
- **Plugin:** Para clasificar el grado de correspondencia en esta categoría es necesario cumplir 3 condiciones:
  - El concepto del enriquecimiento de la tarea de consulta debe pertenecer al nivel 3 del modelo eTOM.
  - El concepto del enriquecimiento de la tarea publicada debe pertenecer al nivel 2 del modelo eTOM.
  - El concepto del enriquecimiento de la tarea de consulta debe hacer parte del conjunto de componentes que conforman el concepto del enriquecimiento de la tarea publicada.
- **Subsume:** Para clasificar el grado de correspondencia en esta categoría es necesario cumplir 3 condiciones:
  - El enriquecimiento de la tarea publicada debe pertenecer al nivel 3 de eTOM
  - El enriquecimiento de la tarea de consulta debe pertenecer al nivel 2 de eTOM.
  - El enriquecimiento de la tarea de consulta debe hacer parte del conjunto de componentes que conforman el enriquecimiento de la tarea publicada.
- **Intersect:** Cuando los conceptos del enriquecimiento semántico de las tareas de consulta y publicada son de nivel 3, y hacen parte del mismo componente de nivel 2.
- **Fail:** Cuando ninguna de las condiciones anteriores se cumple.

Tomando como referencia estas cinco categorías de correspondencia se diseñó un algoritmo para clasificar el grado de correspondencia entre parejas de identificadores de tareas comparadas (Algoritmo 4-4). Según este algoritmo, cuando el concepto de la tarea de consulta (I_Q) es igual al concepto de la tarea publicada (I_T) el grado de correspondencia es establecido como *exact*. De lo contrario se procede a encontrar el nivel al que pertenecen dichos conceptos (LI_Q y LI_T) y se verifica que tanto I_Q como I_T pertenezcan a los niveles 2 o 3 de la misma área funcional. Si esto se cumple se evalúa si tanto el concepto de la tarea de consulta (I_Q) como el de la tarea publicada (I_T) pertenecen al nivel 3, con lo cual la correspondencia se establece como *intersect*. Si no es así se evalúa si el concepto de la tarea publicada (I_T)

pertenece a nivel 2 determinando la correspondencia como *plugin*, o si el concepto de la tarea de consulta (I_Q) pertenece a nivel 2 definiendo la correspondencia como *subsume*. Finalmente, si no se cumple ninguna de las condiciones anteriores entonces se dice que grado de correspondencia es *fail*.

Algoritmo 4-4: Algoritmo de determinación del grado de correspondencia entre parejas de conceptos.

```
1   procedure SemanticMatchmaking (IQ, IT)
2   if IQ  = IT then  matchingDegree := exact
3   else
4   begin
5        LIQ :=getLevel(IQ)
6        LIT := getLevel(IT)
7        if (LIQ ≠ 2  or  LIQ ≠ 3  or  LIT ≠ 2  or  LIT ≠ 3) then
8             matchingDegree := fail
9        else
10       begin
11            SLCQ := getSecondLevelConcept(IQ)
12            {SLCQ is the second level concept of the IQ}
13            SLCT := getSecondLevelConcept(IT)
14            {SLCT is the second level concept of the IT}
15            if (SLCQ = SLCT) then
16                 if (LIQ = 3 and LIT = 3) then matchingDegree
17                      := intersect
18                 else if (LIT = 2) then matchingDegree :=
19                      plugin
20                      else matchingDegree := subsume
21            else
22                 matchingDegree := fail
23       end
24  end
25  La relación entre CQ  y CT está definida por matchingDegree
```

De esta manera el mecanismo establecido para calcular la similitud semántica entre dos conceptos se definió a partir de sus relaciones en cuanto a distancia semántica y grado de correspondencia dados según la ontología. Matemáticamente este mecanismo está representado por la siguiente fórmula:

$$SS = \frac{1}{p * SD + 1}$$

Ecuación 4-4

Donde SD es la medida de la Distancia Semántica entre los dos conceptos en la ontología de dominio, y p es un factor que permite condicionar el impacto de la distancia semántica en el valor de la similitud. Los valores de p se asignaron de la siguiente manera: si el grado de correspondencia es *exact* $p = 0$, si es *plugin* $p = 0.2$, si es *subsume* $p = 0.6$, si es *intersect* $p = 0.8$ y si es *fail* $p = 1$.

A continuación, para la determinación de la similitud semántica entre los identificadores de dos tareas (del BP consulta y del BP repositorio), se utiliza el Algoritmo 4-5, el cual inicia evaluando el grado de correspondencia; si es *exact entonces* p = 0, con lo que la similitud semántica (SS) se establece en 1. De lo contrario, calcula la distancia semántica (denotada por SD) y el grado de correspondencia entre C_Q y C_T (MD). Luego, de acuerdo con el resultado obtenido en MD y dependiendo de la ontología registrada en DO, ajusta el parámetro *p* que interviene en el cálculo de la similitud semántica junto con SD

Algoritmo 4-5: Algoritmo Función Similitud semántica entre conceptos.

```
1   procedure findSemanticSimilarity (CQ, CT, DO)
2   p := 0
3   SD :=0
4   MD := findMatchingDegree (CQ, CT)
5   if MD ≠ exact then
6   begin
7        SD:= findSemanticDistance (CQ, CT)
8        if MD = plugin then
9             if DO = SSID then p := 0.7
10            else p := 0.2
11       else if MD = subsume then
12            if DO = SSID then p := 0.9
13            else p := 0.5
14       else if MD = intersect then
15            if DO = SSID then p := 0.1
16            else p := 0.9
17       else p := 1
18  end
19  CQ es el enriquecimiento de la tarea del BP consulta
    CT es el enriquecimiento de la tarea del BP repositorio
    DO es un concepto de la ontología de dominio
    p es un factor que depende del grado de correspondencia
    SD es la distancia semántica entre CQ y CT
    MD es el grado de correspondencia entre CQ y CT}
```

SS := 1/(p*SD + 1)
SS es la similitud semántica entre C_Q y C_T}

- **Similitud Semántica de interfaces:**

  Debido a que las tareas comparadas pueden contener varias entradas y salidas al mismo tiempo, el mecanismo para calcular la similitud semántica en este caso es más complejo que en el caso de los identificadores. En este proyecto se aborda la comparación de entradas/salidas de acuerdo al trabajo de Benatallah et al [17] en el cual a partir de una petición de servicio, se encuentran las combinaciones de servicios que mejor "cubren" las capacidades requeridas. Dicha combinación de servicios debe compartir el mayor número posible de entradas y salidas con el servicio solicitado sin exceder las entradas del mismo.

  En este sentido la adaptación de este enfoque consiste en determinar una tarea de un BP repositorio (T_T) más similar a una tarea del BP consulta (T_Q), teniendo en cuenta las siguientes condiciones:

  - T_T debe compartir el mayor número posible de salidas con T_Q
  - T_T debe compartir y, en lo posible, no exceder las entradas de T_Q

  Decir que T_T comparte una entrada/salida con T_Q, no implica necesariamente que dicha entrada/salida sea la misma para ambas tareas, es decir, estas entradas/salidas pueden ser sintácticamente diferentes pero semánticamente similares. De esta manera el mecanismo propuesto para calcular la similitud semántica de las entradas/salidas de dos tareas, consiste en evaluar en primera instancia las similitudes individuales entre parejas de entradas/salidas, y posteriormente analizar esos resultados para cumplir con las dos condiciones establecidas. Este mecanismo se explica a continuación para el caso de las entradas, considerando que para las salidas se aplica el mismo procedimiento pero con algunas variaciones

- **Similitud Semántica de Entradas**

La similitud entre parejas de entradas de una T_T y una T_Q, se calcula a partir del mecanismo para la comparación de conceptos explicado en el Algoritmo 4-5. Posteriormente los resultados se organizan en una matriz de similitud, cuyas filas corresponden a las entradas de la T_Q y las columnas a las entradas de la T_T.

Dicha matriz se analiza para determinar las *similitudes máximas* entre parejas de entradas de las dos tareas comparadas, las cuales hacen referencia a los mayores valores de similitud que pueden obtenerse entre todas las posibles combinaciones de parejas de entradas, garantizando que cada entrada de la T_Q esté relacionada con una única entrada de la T_T y viceversa.

De esta manera, si se define S como la matriz de similitud de entradas de dimensión mxn (donde *m* hace referencia a las entradas de la T_Q y *n* a las de la T_T) y se determina el conjunto de *similitudes máximas* como la agrupación de similitudes S_{ij} que cumplen con la siguiente propiedad:

$$S_{ij} : \forall S_{kj}\ (k = 0,1,2,\dots,m)\ S_{ij} > S_{kj} \wedge \forall S_{il}\ (l = 0,1,2,\dots,n)\ S_{ij} > S_{il}$$

Ecuación 4-5

Una vez definido este conjunto de *similitudes máximas*, se procede a encontrar las entradas que quedaron aisladas (entradas perdidas) de las parejas establecidas en estas relaciones, las cuales se presentan de *m a n* o viceversa. A partir de la especificación de estos parámetros, y teniendo en cuenta las condiciones del análisis de cobertura descritas al inicio de esta sección, se define la Similitud Semántica entre entradas o salidas (SS_{IO}) como:

$$SS_{IO} = \frac{\sum S_{ij}}{\min(m,n) + \beta * |m - n|}$$

Ecuación 4-6

Dónde S_{ij} representa a cada *similitud máximas,* $\min(m,n)$ es el máximo número de parejas de entradas que puede establecerse entre las dos tareas comparadas según la propiedad de las S_{ij} y $\beta * |m-n|$ es un parámetro que permite condicionar el valor de la similitud cuando existen entradas perdidas. En este caso si T_Q tiene más entradas que la T_T entonces β toma el valor de 0.8, de lo contrario toma el valor de 1.

A partir de este mecanismo se definió un algoritmo para calcular la similitud semántica de entradas (Algoritmo 4-6), el cual recibe como parámetros las dos tareas a comparar (T_Q, T_T).

Este algoritmo opera de la siguiente manera, en primer lugar extrae las entradas de la T_Q en el arreglo I_Q [ ] y asigna a *m* el número de entradas de dicha tarea. De la misma manera extrae las entradas de T_T para almacenarlas en I_T [ ] y guarda el tamaño de este arreglo en *n*. Posteriormente obtiene el enriquecimiento de las entradas de T_Q en el arreglo $IE_Q[m]$, y de la T_T en el arreglo $IE_T[n]$.

Con el fin de verificar si existen entradas a comparar, se evalúa si los arreglos I_Q [ ] e I_T [ ] están vacíos, es decir si ninguna de las tareas comparadas tiene entradas, con lo cual se establece que la similitud de entradas (SS_{IO}) es igual a 1. De lo contrario se examina si alguno de ellos está vacío (si cualquiera de las dos tareas no tiene entradas), con lo cual SS_{IO} se hace igual a 0, puesto que si alguna de las tareas tuviera entradas y la otra no entonces no sería posible realizar la comparación en cuanto a este parámetro. Si ninguna de las anteriores condiciones se presenta, se procede a obtener la matriz de similitudes de entradas ($ISM\ [m][n]$), a partir de la comparación de los arreglos de enriquecimiento ($IE_Q[m]$, e $IE_T[n]$,). Luego se analiza esta matriz para obtener las máximas similitudes (MSA [ ]). Una vez reunida esos resultados, se evalúa si la T_Q tiene más entradas que la T_T, en cuyo caso se calcula la similitud semántica de entradas a partir de la Ecuación 4-6.

Algoritmo 4-6: Algoritmo para Calculo similitud semántica de entradas.

```
1   procedure findSemanticInputsSimilarity (TQ, TT)
2   begin
3   IQ [ ] := getInputs(TQ)
4   m := sizeof(IQ [ ])
5   IT [ ] := getInputs(TT)
6   n := sizeof(IT [ ])
7   IEQ [m]:= getInputsEnrichment(TQ)
8   IET [n]:= getInputsEnrichment(TT)
9   if (IQ [ ] = ø and IT [ ] = ø) then SSIO := 1
10  else if (IQ [ ] = ø or IT [ ] = ø) then SSIO := 0
11       else
12       begin
13       ISM [m][n] := getInputSimMatrix(IEQ [m], IET [n])
14       MSA [ ] := getMaxSimsArray(ISM [ ][ ])
15       MSS := addMaxSims(MSA [ ])
16       if (m>n) then  SSIO := MSS /( n + 0.8*|m-n|)
17       else  SSIO = MSS / (m + |m-n|)
18       end
19  end

    TQ es la tarea del BP consulta
    TT es la tarea del BP repositorio
    IQ [ ] es el arreglo de entradas de la TQ
    IT [ ] es el arreglo de entradas de la TT
    IEQ [ ] es el arreglo de enriquecimiento sobre el
    arreglo IQ
    IET [ ] es el arreglo de enriquecimiento sobre el
    arreglo IT
    ISM [m][n] es la matriz de similitudes de entradas
    MSA [ ] es el arreglo de similitudes máximas.
    MSS es la suma de similitudes máximas
    SSIO es el valor de similitudes de entradas entre TQ
    y TT
```

- **Similitud Semántica de Salidas:** el algoritmo para calcular la similitud semántica de salidas entre las dos tareas comparadas, funciona de la misma manera que el algoritmo de las entradas. La única variación de este procedimiento respecto al anterior consiste en modificar el valor de β, el cual cuando T_T, excede el número de salidas de la T_Q toma el valor de 0.8 y en caso contrario el valor de 1, esto permite condicionar el impacto de las salidas perdidas en el valor de similitud. Además se debe reemplazar los arreglos de entradas I_T [ ], I_Q [ ], $IE_Q[m]$, $IE_T[n]$, la matriz *ISM* (matriz de similitudes entre

entradas) por sus correspondientes para salidas O_T [ ], O_Q [ ], $OE_Q[m]$, $OE_T[n]$ y la matriz *OSM* (matriz de similitudes entre salidas).

Una vez obtenidos los valores de similitud de identificadores, entradas y salidas mediante los algoritmos explicados anteriormente, se procede a calcular la similitud semántica general (OSS) entre las dos tareas comparadas, a través de una suma ponderada de dichos resultados (Ecuación 4-7):

$$OSS = SS * wId + SS_I * wIn + SS_O * wO$$

Ecuación 4-7

Donde SS es la similitud semántica de identificadores (Ecuación 4-4); wId es un porcentaje que determina la contribución de la similitud de identificadores en la similitud semántica general de las tareas; $\mathrm{SS_I}$ es la similitud semántica de Entradas (Ecuación 4-6); wIn es el porcentaje que determina la contribución de la similitud de entradas en la similitud semántica general de las tareas; $\mathrm{SS_O}$ es la similitud semántica de Salidas (Ecuación 4-6); wO es el porcentaje que determina la contribución de la similitud de identificadores en la similitud semántica general de las tareas, y los valores de wId, wIn y wO son parámetros de configuración son proporcionados por los usuarios.

Finalmente, la distancia de nodo tipo tarea se puede calcular utilizando el Algoritmo 4-7, el cual a su vez utiliza los algoritmos descritos para la similitud semántica de identificadores (Algoritmo 4-5) y la similitud semántica de interfaces (Algoritmo 4-6).

Algoritmo 4-7: Algoritmo para la función *TaskCost*

```
1   INPUTS: (T_Q, T_T)
2   OUTPUT: tnd
3   if T_Q type ≠ T_T type then
4      return tnd = 1
5   else if T_Q type = FunctionType then
6          if T_Q y T_T tienen enriquecimiento Semántico
7             SS = findSemanticSimilarity (T_Q, T_T, DO) (Algoritmo 4-5)
8             SS_I = findSemanticInputSimilarity (T_Q, T_T) (Algoritmo 4-6)
9             SS_O = findSemanticOutputSimilarity (T_Q, T_T) (Algoritmo 4-6)
10            TSS = SS + SS_I + SS_O
11
```

```
12          return tnd = 1 − TSS
13        else
14           LS = findLingüisticSimilarity(T_Q,T_T) (Ecuación 4-3)
15
16        return tnd = 1 - LS
17        end if
18  end if T_Q type = EventType then
19           LS = findLingüisticSimilarity(T_Q,T_T) (Ecuación 4-3)
20        return tnd = 1 − LS
21  end if
```

tnd es la distancia de nodo tipo tarea
SS es la similitud semántica entre conceptos relativos a los identificadores de las tareas T_Q.y T_T
DO es una ontología de dominio
SS_I es la similitud semántica entre entradas de las tareas T_Q.y T_T
SS_O es la similitud semántica entre salidas de las tareas T_Q.y T_T
TSS es la similitud semántica total entre dos tareas T_Q.y T_T
LS es la similitud léxica entre dos identificadores de las tareas T_Q.y T_T.

4.4 Resultados de Similitud

Este módulo (Figura 4-1(d)) evalúa la similitud entre un grafo de consulta y los grafos obtenidos del repositorio utilizando los costos de edición presentados en las secciones anteriores. En este caso las funciones de similitud se han clasificado de acuerdo a tres características de los BP: estructura, lingüística y comportamiento secuencial.

4.4.1 Similitud Estructural (*Stsim*)

Calcula la similitud como función de la distancia de edición total (TD) encontrada por el algoritmo de correspondencia estructural.

$$Stsim = \frac{1}{1 + TD}$$

Ecuación 4-8

4.4.2 Similitud Lingüística de Nodo (*LNsim*)

Retorna un valor de similitud calculado como función de la similitud estructural, el número de nodos en el grafo resultante que corresponden al grafo de consulta (*nodos intersectados*) y el número total de nodos en el grafo de consulta (*nodos de consulta*) de acuerdo a la comparación lingüística presentada en la sección 4.3.

$$LNsim = \frac{Stsim * nodos\ intersectados}{nodos\ consulta}$$

Ecuación 4-9

4.4.3 Similitud de Comportamiento Secuencial (*SBsim*)

Estima la similitud como función de la similitud estructural y el número de secuencias compuestas por *n* nodos (n-secuencias). Esta medida de similitud relaciona las n-secuencias del proceso de consulta (*nsSeqQ*), el proceso obtenido del repositorio (*nSeqT*) y el grafo resultante (nSeqT).

$$SBsim = \frac{Stsim}{1 + nSeqQ + nSeqT - 2 * nSeqT * nSeqQ}$$

Ecuación 4-10

Utilizando estas tres medidas de similitud se clasificaron los BP obtenidos del repositorio en tres listas ordenadas desde el más similar hasta el menos similar respecto a un grafo de consulta. Esas listas ordenadas se presentan al usuario final a través del módulo organizador de ranking estructural (Figura 4-1(e)), el cual tiene una interfaz gráfica de usuario que facilita a los usuarios visualizar los resultados de similitud, las operaciones de edición ejecutadas y el grafo de resultado en comparación con el grafo de consulta.

4.5 Resumen

En este capítulo se presentó el módulo analizador estructural y semántico el cual ejecuta un análisis de correspondencia estructural entre un BP consulta y un conjunto de BP repositorio (cada BP de la lista ordenada obtenida del repositorio). Este análisis realiza una serie de operaciones de edición sobre cada uno de los BP repositorio con el fin de hacerlos tan similares como sea posible a un BP consulta. Como resultado se entrega un ranking de grafos de resultado y su correspondencia con el BP consulta.

Capítulo 5

5 Resultados y Discusión

Este capítulo presenta los materiales y métodos utilizados en esta propuesta para evaluar experimentalmente el entorno de recuperación de BP basado en semántica del comportamiento y discute los resultados obtenidos. La evaluación experimental se realizó en dos análisis; el primero evalúa el rendimiento del prototipo y el segundo su relevancia. Los resultados de relevancia se utilizaron para comparar las medidas de precisión y exhaustividad del prototipo BeMantics con un trabajo previo denominado BeMatch (A Platform for Matchmaking Service Behavior Models) desarrollado por Corrales et al [113].

5.1 Conjunto de procesos de negocio para pruebas

Para probar el entorno de recuperación de BP basado en semántica de comportamiento, en esta propuesta se desarrolló un conjunto de BP (conjunto de prueba) utilizando el lenguaje de modelado BPMO. El conjunto está compuesto por 60 BP del dominio de telecomunicaciones y 40 BP del dominio de procesamiento. En este caso, solo los BP de telecomunicaciones se enriquecieron semánticamente utilizando las ontologías de dominio seTOM y sSID. En el Anexo B.4 se presentan los modelos eTOM y SID, en el Anexo C el procedimiento de enriquecimiento semántico y en el Anexo F información acerca del conjunto de BP utilizados en esta propuesta.

5.2 Modelo de Evaluación de Pertinencia

Para evaluar el entorno presentado en este trabajo en términos de relevancia de BP se diseñó un modelo de evaluación de pertinencia [114] el cual permite que un conjunto de 6 jueces humanos emita juicios de relevancia entre los BP del conjunto de prueba y 6 BP actuando como consultas. Por lo tanto, este modelo es útil para catalogar herramientas de descubrimiento de BP como efectivas si sus resultados corresponden con evaluaciones de similitud de los jueces, las cuales entregan como resultado un conjunto de BP relevantes. En esta propuesta el modelo de evaluación fue implementado a través de una plataforma Web denominada *"Herramienta de Evaluación de Pertinencia"* la cual se presenta en detalle en el Anexo G.

5.2.1 Criterios de evaluación.

Los criterios de evaluación proporcionan valores numéricos para facilitar que los seres humanos emitan juicios de semejanza entre los BP con el fin de obtener conjuntos jerarquizados de relevancia confiables. El modelo descrito en el presente trabajo permitió clasificar los criterios para la evaluación intuitiva en cuatro niveles analizados comúnmente en las herramientas automáticas: Estructura, Comportamiento, Semántica e Interfaces, cada uno de estos niveles se valora utilizando juicios de semejanza para el emparejamiento de BP establecidos en trabajos previos [23, 115, 116].

Estructura: este nivel analiza la representación secuencial de las tareas que componen un BP y sus relaciones. Los criterios para su evaluación son los siguientes:

- *Dependencia Causal:* evalúa la semejanza de dos BP teniendo en cuenta relaciones de dependencia entre las tareas que los componen.

- *Estructura gráfica:* brinda una idea visual de la secuencia de las tareas dentro de un BP. Para la evaluación de este criterio se estableció la siguiente escala:

exacto cuando gráficamente el BP consulta y el BP repositorio son iguales; *complemento* cuando el BP repositorio está contenido, es parte o hereda estructuras del BP consulta; *inclusión* cuando el BP consulta está contenido, es parte o hereda del BP repositorio; *inexacta* cuando el BP repositorio presenta estructuras que no son exactamente iguales al BP consulta, pero que guardan alguna relación; y *fallida* cuando estructuralmente no existe relación alguna que sea común entre tareas de los BP comparados.

***Comportamiento*:** este nivel tiene en cuenta el flujo de control de las tareas dentro del BP, es decir, los constructores específicos que determinan el funcionamiento de un BP de acuerdo a la ejecución de sus tareas [25]. Para evaluar este nivel el criterio que mejor se adapta es el flujo de control, el cual permite representar el orden en el cual se ejecutan las tareas para cumplir con el objetivo del BP.

Semántica: este nivel analiza los conceptos propios de las tareas de acuerdo a un dominio específico de aplicación [16]. Estos conceptos corresponden a los nombres de las tareas y las descripciones verbales de sus funcionalidades. En este caso se evaluaron los siguientes criterios:

- *Nombres:* corresponde a nombres o etiquetas de las tareas de cada uno de los BP.

- *Descripciones:* proporciona una descripción textual del funcionamiento de las tareas.

Interfaces: este nivel evalúa datos y tipos de datos de las entradas y salidas de tareas pertenecientes a dos BP. Los criterios para evaluar este nivel son los siguientes:

- *Entradas:* permite comparar las listas de entradas de las tareas de dos BP, es decir, busca si para cada tarea de un BP consulta existe una y sólo una tarea de un BP repositorio cuya lista de entradas posea elementos semejantes.

- *Salidas:* este criterio evalúa las salidas de los BP de igual manera que el criterio anterior.

En el presente proyecto todos los criterios presentados se evaluaron en una escala numérica con la asignación de valores desde 0 Completamente diferentes a 4 Idénticos [54].

5.2.1 Medidas de recuperación de los BP.

Las medidas de recuperación evalúan los criterios en términos del rendimiento y la relevancia de recuperación [117-120]. El rendimiento está relacionado con el tiempo de respuesta; y la relevancia con la efectividad de recuperación, la cual se puede estimar utilizando las medidas de precisión (P) y la exhaustividad (R). La precisión evalúa la habilidad del sistema para recuperar solamente elementos relevantes (aquellos elementos considerados como similares a consultas por los jueces) y evitar resultados inesperados o falsos positivos (es decir, elementos recuperados no relevantes). Y la exhaustividad evalúa la habilidad para recuperar todos los elementos relevantes evitando perder resultados relevantes (es decir falsos negativos).

En esta propuesta se utilizó un modelo gradado de relevancia (Pg y Rg) [121] la cual considera diferentes niveles de relevancia en contraste con la relevancia binaria tradicional la cual solo tiene en cuenta dos niveles (relevantes y no relevantes) [118]. De esta manera se estimó la relevancia de recuperación del entorno BeMantics comparando un BP de consulta (Q) con cada elemento ($\mathbf{T_i}$) de un conjunto de BP obtenidos del repositorio. BeMantics entrega una lista ordenada automática ($\mathbf{f_e}$) y la herramienta de evaluación de pertinencia entrega una lista ordenada real ($\mathbf{f_r}$) generada por los juicios de evaluación de los jueces humanos. Por lo tanto las medidas Pg y Rg se pueden calcular utilizando las siguientes ecuaciones.

$$Rg = \frac{\sum_{T_i \epsilon T} \min\{f_r(Q, T_i), f_e(Q, T_i)\}}{\sum_{T_i \epsilon T} f_r(Q, T_i)}$$

Ecuación 5-1

$$Pg = \frac{\sum_{T_i \epsilon T} \min\{f_r(Q, T_i), f_e(Q, T_i)\}}{\sum_{T_i \epsilon T} f_e(Q, T_i)}$$

Ecuación 5-2

5.3 Prototipo Experimental

El prototipo experimental desarrollado consta de dos módulos principales de acuerdo al modelo BeMantics propuesto en este trabajo. El primero es el repositorio de BP basado en semántica del comportamiento y el segundo el analizador estructural y semántico. A continuación se describen los casos de uso del sistema BeMantics en general

5.3.1 Diagrama de Casos de Uso de BeMantics

En la Figura 5-1 se pueden observar los principales casos de uso del sistema BeMantics, incluyendo las funcionalidades del repositorio (almacenar BP, Recuperar modelos BP y Gestionar Modelos BP) y las funcionalidades del analizador estructural (ejecutar correspondencia estructural, ver resultados correspondencia)

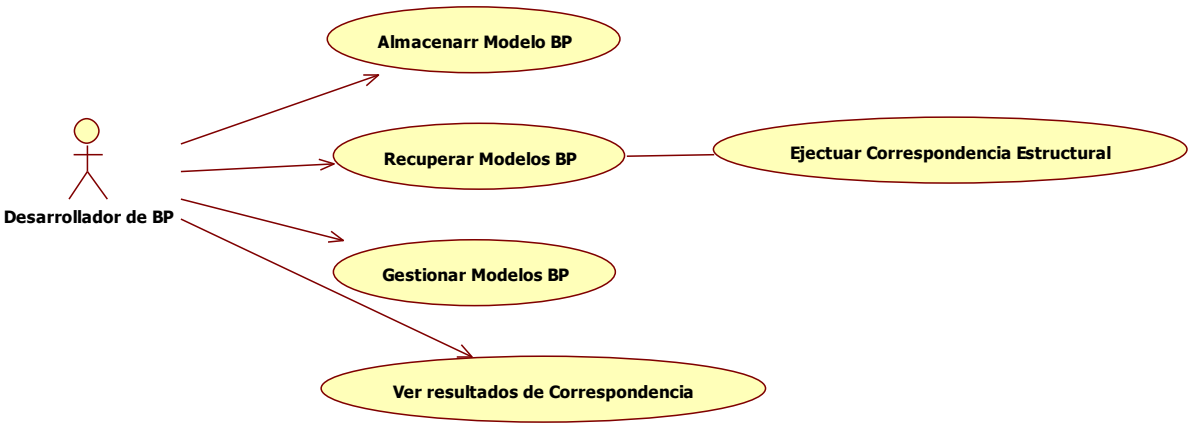


Figura 5-1: Modelo de casos de uso para el repositorio

Caso de Uso No. 1: *Almacenar Modelo BP.*
Iniciador: Desarrollador de BP (Agente Externo: Persona, Aplicación)
Propósito: Detectar patrones de flujo de control al interior de un BP y almacenar el BP junto con los patrones detectados. .
Resumen: Este caso de uso permite detectar el conjunto de patrones (de los 12 patrones básicos) que un BP contiene, con el fin de almacenarlo en el repositorio y etiquetarlo con el conjunto de patrones detectados.
Interfaz Gráfica: la interfaz gráfica de este caso de uso solicita el ingreso de un proceso de negocio previamente desarrollado en WSMOStudio. Como se puede observar en la Figura 5-2 se solicita el ingreso de un proceso BPMO enriquecido semánticamente.  Figura 5-2: GUI de almacenamiento de un BP en el repositorio

Caso de Uso No. 2: *Recuperar Modelos BP*
Iniciador: Desarrollador de BP (Agente Externo: Persona, Aplicación)
Propósito: Entregar un conjunto de Modelos BP ordenados según la medida de similitud total de su flujo de control, con respecto a un Modelo BP de consulta.
Resumen: Este caso de uso permite entregar una lista ordenada de BP del repositorio a partir de los patrones de control de flujo detectados en un BP de consulta.

Caso de Uso No. 3: *Ejecutar Correspondencia Estructural*
Iniciador: Desarrollador de BP (Agente Externo: Persona, Aplicación)
Propósito: Si la persona lo decide, puede refinar los resultados de la recuperación de BP ofrecida por el repositorio. Para esto puede ejecutar un analizador estructural y semántico, el cual utiliza un algoritmo de isomorfismo de grafos basado en corrección de errores con el fin de encontrar correspondencias y similitudes entre un BP de consulta y un conjunto de BP entregados por el repositorio.
Resumen: Este caso de uso permite entregar una lista de correspondencias entre un BP consulta y una lista ordenada de BP repositorio, basándose en criterios de estructura y semántica aplicada a las tareas de los BP.
Interfaz Gráfica: En este caso de uso primero se deben seleccionar los parámetros a utilizar para ejecutar la correspondencia (Figura 5-3) y luego hacer

clic en el botón “*Execute Matching*”.

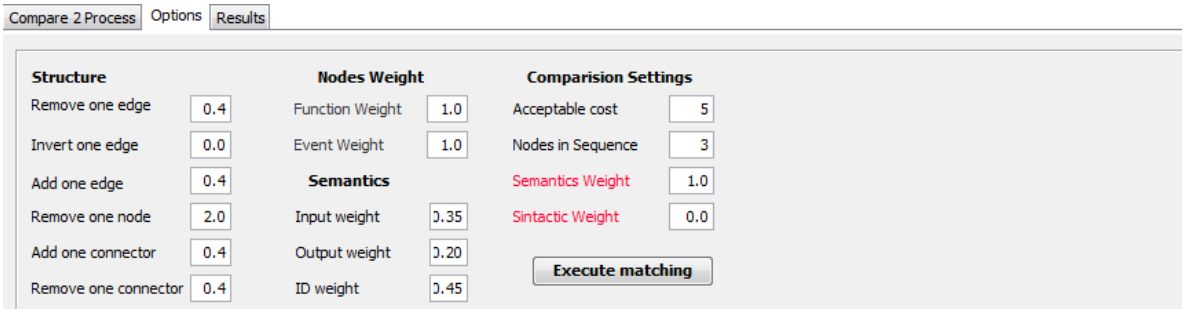


Figura 5-3: GUI de selección de parámetros y ejecución de Correspondencia

Caso de Uso No. 4: *Gestionar Modelos BP.*
Iniciador: Desarrollador de BP (Agente Externo: Persona, Aplicación)
Propósito: Permitir la ejecución de funciones CRUD (Create, Read, Update y Delete) sobre los BP almacenados en el repositorio.
Resumen: Este caso de uso permite a los usuarios visualizar, editar, y eliminar los BP almacenados en el repositorio.

Caso de Uso No. 5: *Ver Resultados Correspondencia*
Iniciador: Desarrollador de BP (Agente Externo: Persona, Aplicación)
Propósito: Permite al usuario observar los resultados de correspondencia que BeMantics propone para un BP consulta y cada uno de los BP repositorio. Además presenta las operaciones de edición ejecutadas en cada BP repositorio para hacerlo similar al BP consulta.
Resumen: Este caso de uso permite visualizar los resultados del prototipo BeMantics.
Interfaz Gráfica: la interfaz gráfica de este caso de uso muestra los resultados de correspondencia estructural, es decir, los valores de las operaciones de edición ejecutadas y la correspondencia entre cada uno de los BP repositorio y el BP de consulta (Figura 5-4).

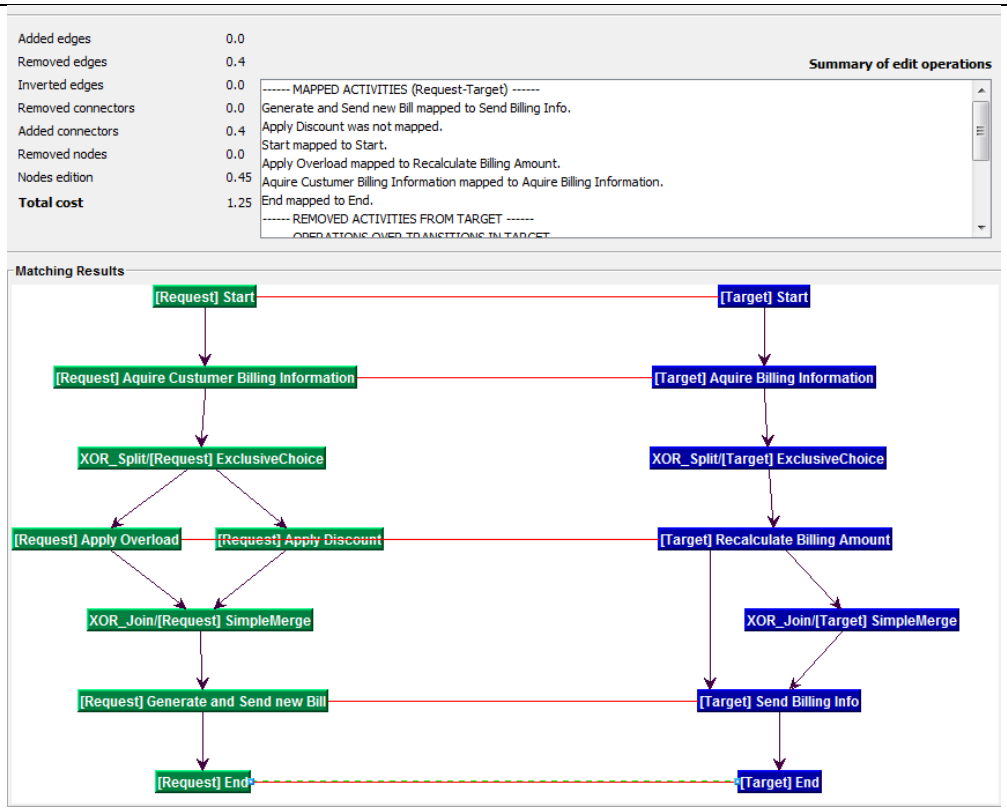


Figura 5-4: GUI de resultados de correspondencia

5.3.2 Diagrama de Paquetes del repositorio

La arquitectura lógica del repositorio está implementada en 3 capas diferentes (**¡Error! No se encuentra el origen de la referencia.**: Aplicación (a), Mediación (b), Fundamentación (c)) en las cuales se presentan las interacciones entre los paquetes más relevantes que componen cada capa. A continuación se presenta el diagrama de paquetes para el repositorio en la **¡Error! No se encuentra el origen de la referencia.**.

Capa de Aplicación: implementa las funcionalidades principales del repositorio como detectar patrones de control de flujo, almacenar BP y ordenar los BP del repositorio de acuerdo a los patrones de un BP consulta.

- **Diseñador BP:** corresponde a una interfaz gráfica a través de la cual el usuario diseña un proceso en el lenguaje BPMO.

- **Transformador BPMO – Grafos:** este paquete contiene las clases necesarias para transformar un BP descrito en WSML a un grafo de proceso y posteriormente a un grafo TD.
- **Detector de Patrones:** corresponde al paquete encargado de detectar un conjunto de patrones dentro de un grafo TD a partir de un conjunto de 12 patrones básicos.
- **Publicador BP:** este módulo permite almacenar un BP en el repositorio, incluyendo sus patrones detectados.
- **Repositorio de Archivos BPMO:** en este repositorio se almacenan los archivos de los procesos BPMO los cuales están en el formato WSML.
- **Repositorio de procesos TD:** corresponde al repositorio en el cual se almacenan los grafos TD de los BP, los cuales se almacenan en una base de datos Berkeley. Es en este tipo de grafos en los que se detectan los patrones.
- **Relaciones Patrones – BP:** este paquete corresponde a la base de datos que almacena las rutas de los archivos BPMO, los relaciona con la ruta de los grafos TD correspondientes y almacena referencias a los patrones detectados.
- **Organizador Semántico de Resultados:** clases encargadas de ejecutar una consulta SQL en la cual se busca una lista de BP repositorio que contengan un conjunto de patrones detectados en un BP consulta. Luego esta lista se ordena en orden descendente de acuerdo al grado de similitud.

Capa de Mediación: corresponde a librerías que permiten al repositorio ejecutar funcionalidades ya implementadas por otros trabajos y acceder a la capa de fundamentación. .

- **wsmo4j:** es un conjunto de librerías que permiten leer archivos WSML y extraer un proceso BPMO para convertirlo a un modelo de objetos en Java.
- **WSMO2Reasoner:** es un conjunto de librerías que calcular las diferentes relaciones que existen entre los conceptos de una ontología descrita con el lenguaje WSML.
- **JDBC:** es el conector para base de datos de Java, el cual permite administrar la base de datos Patrones – BP utilizando el lenguaje de programación Java.

Capa de Fundamentación: en esta capa se encuentran las tecnologías base sobre las cuales se ejecuta el prototipo.

- **JDK:** corresponde al entorno de desarrollo de aplicaciones Java.
- **PostgreSQL:** es un motor de administración de bases de datos relacionales, en el cual permite administrar la base de datos de relaciones Patrones – BP.
- **Berkeley DB:** es una base de datos desarrollada por Oracle, la cual permite almacenar grafos TD.
- **Ontología de Patrones:** es una ontología que se creó en este proyecto utilizando el lenguaje WSML. Esta ontología contiene diferentes relaciones entre patrones de control de flujo.
- **WSML:** es un lenguaje de modelado semántico, el cual permite modelar servicios web, procesos de negocio y ontologías.
- **BPMO:** es un lenguaje de modelado semántico de procesos de negocio que se fundamenta en la base del lenguaje WSML.
- **Ubuntu 10.04:** es el sistema operativo que soporta al entorno BeMantics.

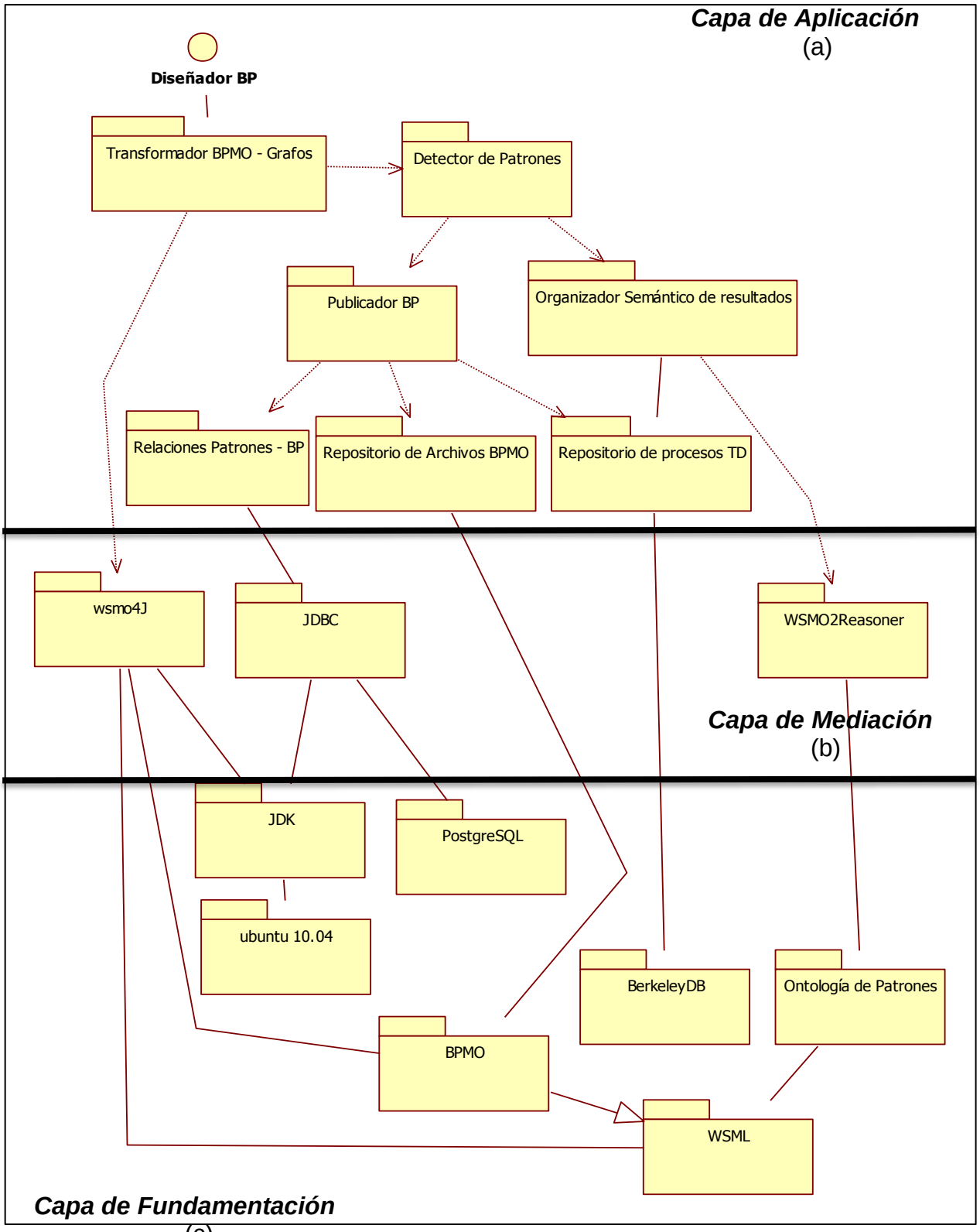


Figura 5-5: Diagrama de paquetes del repositorio.

5.3.3 Diagrama de Paquetes del Analizador Estructural.

Al igual que el repositorio la arquitectura lógica del repositorio está implementada en 3 capas diferentes (Aplicación, Mediación, Fundamentación), las cuales se presentan en la Figura 5-5 (a, b y c).

Capa de Aplicación:

- **Diseñador de BP Consulta:** permite diseñar gráficamente un BP utilizando el lenguaje de modelado BPMO.
- **Repositorio de BP:** este paquete corresponde al módulo anterior, presentado en la sección 5.3.2. El cual recibe el proceso BPMO lo transforma a proceso TD y retorna una lista de BP repositorio que cumplen con la condición de contener un conjunto de patrones similar al conjunto de patrones detectados en un BP consulta.
- **Analizador de Correspondencia estructural:** es el paquete que contiene las clases para ejecutar la correspondencia estructural y semántica entre un BP consulta y un conjunto de BP repositorio a través de un algoritmo de isomorfismo de sub-grafos con corrección de error. .
- **Analizador lingüístico:** es el módulo encargado de determinar la similitud léxica o semántica (según sea el caso, si la tarea tiene o no enriquecimiento semántico) entre dos tareas de tipo función o evento.
- **Analizador Léxico:** es el encargado de encontrar una distancia léxica entre los nombres de una tarea de tipo evento o de una tarea tipo función sin enriquecimiento semántico.
- **Analizador Semántico:** encargado de encontrar una similitud semántica entre dos tareas de tipo función las cuales contienen enriquecimiento semántico tanto en su identificador (nombre) y sus interfaces (entradas/salidas).
- **Calculador de Funciones de Costo:** este método permite calcular la distancia entre dos BP, estimando un costo total de las operaciones de edición ejecutadas por el analizador de correspondencia estructural.
- **Calculador de Funciones de Similitud:** calcula las funciones de similitud a partir de las funciones de costo.
- **Organizador Estructural:** toma los valores de las funciones de similitud y organiza los BP repositorio de mayor a menor similitud.

- **Vista de Resultados:** permite al usuario final la visualización de los resultados de correspondencia entre el BP consulta y cada uno de los BP repositorio, además de las operaciones de edición ejecutadas.

Capa de Mediación:

- **WSMO2Reasoner:** es un conjunto de librerías que calcular las diferentes relaciones que existen entre los conceptos de una ontología descrita con el lenguaje WSML.
- **JGraph:** es una librería que permite dibujar grafos bajo el lenguaje Java.
- **JGraphLayout:** permite posicionar los grafos dibujados con JGraphLayout en una interfaz gráfica.

Capa de Fundamentación:

- **WordNet:** es una base de datos léxica que contiene relaciones léxicas entre cadenas de texto.
- **Ontologías de Dominio:** corresponden a ontologías del dominio de las telecomunicaciones las cuales contienen un conjunto de conceptos relacionados que permiten realizar inferencias semánticas. Las ontologías de dominio utilizadas en este trabajo de grado corresponden a seTOM (para los identificadores de las tareas) y sSID (para las entradas y salidas).
- **WSML:** es un lenguaje de modelado semántico, el cual permite modelar servicios web, procesos de negocio y ontologías.
- **Ubuntu 10.04:** es el sistema operativo que soporta al entorno BeMantics.
- **JDK:** corresponde al entorno de desarrollo de aplicaciones Java

5.4 Análisis de Rendimiento

El análisis de rendimiento evalúa la velocidad de respuesta en tiempo del prototipo BeMantics considerando sus dos módulos principales el repositorio y el analizador de semántica del comportamiento. En los dos casos el tiempo de respuesta se estimó teniendo en cuenta el número de nodos de los BP repositorio y los tiempos respectivos al almacenamiento y generación de la lista ordenada de BP

repositorio. El conjunto total de resultados del análisis de rendimiento se pueden encontrar en el Anexo I.2.

Por otra parte, el servidor de prueba sobre el cual se ejecutaron los análisis de rendimiento tiene las características hardware y software presentadas en la Tabla 5-1.

Característica	**Valor**
RAM	4 GB
Procesador	Intel i3-530 (2.93 GHz)
Sistema Operativo	Linux Ubuntu 9.10
Lenguaje de Programación	J2SDK 1.6
Entorno de Desarrollo	Netbeans 6.9
Gestor de Bases de Datos	PostgreSQL 8.4

Tabla 5-1: Características del servidor de prueba

5.4.1 Repositorio de BP basado en semántica del comportamiento.

Para analizar la ejecución del repositorio se almacenó un conjunto de 100 BP y se probó el sistema de clasificación con 6 procesos de consulta. La Figura 5-6 muestra la evaluación del almacenamiento de BP en el repositorio de acuerdo al su número de nodos.

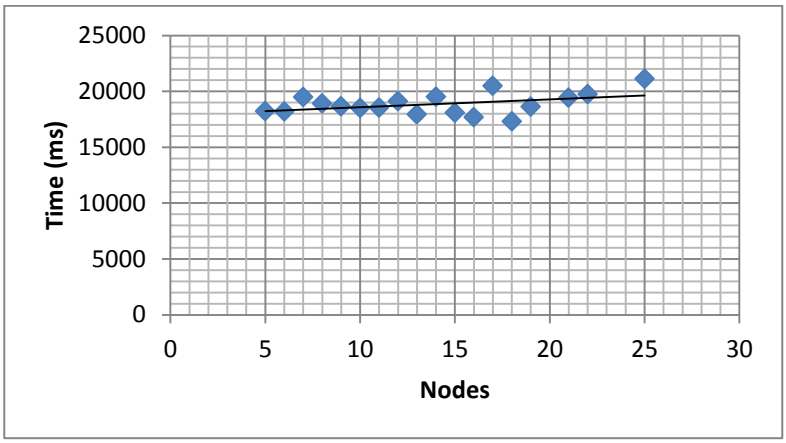


Figura 5-6: Evaluación del rendimiento del almacenamiento de un BP en el repositorio de acuerdo a su número de nodos

En esta gráfica se puede observar que el tiempo consumido para almacenar un BP en el repositorio mantiene valores que varían entre los 15 a 25 segundos, y que tiene una tendencia lineal de crecimiento en dependencia del número de nodos de cada BP.

Por otra parte, la Figura 5-7 muestra la evaluación de rendimiento del procedimiento de recuperación de BP del repositorio de acuerdo a la semántica del comportamiento.

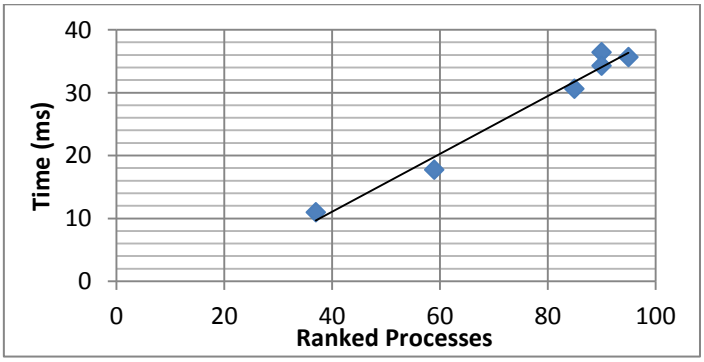


Figura 5-7: Evaluación del rendimiento de recuperación de BP ejecutada en el repositorio

Los resultados muestran que el tiempo consumido para generar una lista ordenada de 40 a 90 BP estuvo entre 10 a 40ms, este tiempo tan bajo se debe a que en el repositorio cuando se recuperan BP de acuerdo a los patrones de control de flujo, se hace simplemente una consulta a una base de datos relacional que tiene ya definidos los patrones de control de flujo que cada BP repositorio contiene. Esto es así, dado que la detección de patrones de control de flujo para los BP repositorio se realiza en la fase de almacenamiento y en la fase de recuperación solo se hace para el BP consulta.

5.4.2 Analizador semántico y estructural.

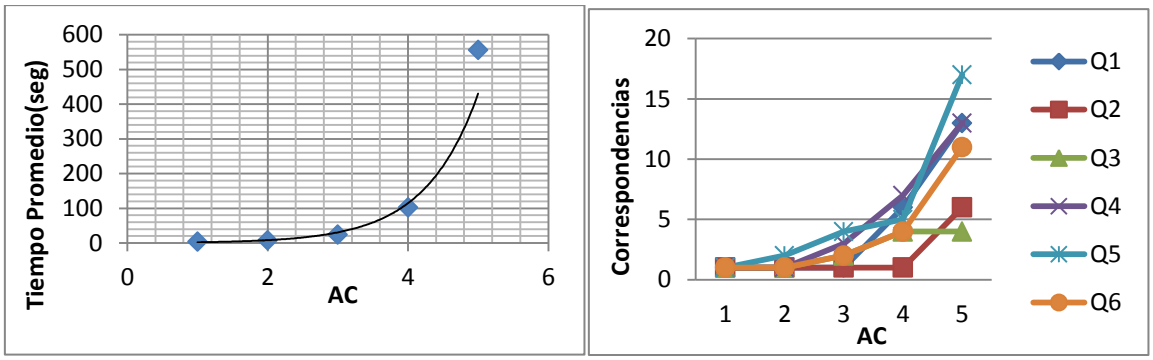
Como se presentó en la sección 4.1 el analizador estructural y semántico puede ser personalizado por los usuarios utilizando los parámetros de ejecución. Sin embargo, para fines de evaluación, dichos parámetros fueron seleccionados arbitrariamente tal como se muestra en la Tabla 5-2.

Parámetro de Ejecución	**Valor**
Eliminar Arista (ϑ)	0,5
Insertar Arista (μ)	0,5
Eliminar Conector (ρ)	0,5
Insertar Conector (σ)	0,5
Eliminar Nodo (τ)	1,0
Substituir Nodo (φ)	0,7

Tabla 5-2: Valores para los parámetros de ejecución del analizador estructural

Adicionalmente, se definió un parámetro denominado *Costo aceptable (AC) (para* determinar el costo máximo para las operaciones de edición) el cual se seleccionó experimentalmente. En este caso se probaron diferentes valores de AC con el fin de determinar un valor apropiado para encontrar un buen número de correspondencias en un tiempo aceptable. En la Figura 5-8(a) se puede observar la evaluación de rendimiento del analizador estructural y semántico con diferentes valores de AC, y la Figura 5-8(b) muestra los valores de AC y el número de correspondencias obtenidas

con los 6 procesos de consulta (Q_1, …, Q_6). En los resultados de la Figura 5-8 se puede concluir que 5 es el mejor valor para AC dado que permitió encontrar un consumo intermedio de tiempo y un buen número de correspondencias.



(a) (b)

Figura 5-8: Selección del parámetro Costo Aceptable (AC): (a) De acuerdo al consumo de tiempo consumido, (b) De acuerdo al número de correspondencias.

Además, la Figura 5-8(a) exhibe un comportamiento exponencial, lo cual indica que un pequeño incremento en el valor de AC puede representar un gran incremento en el tiempo de consumo; y la Figura 5-8 (b) muestra que para el valor 5 se obtuvieron valores altos de correspondencias para todas las consultas. No obstante, para valores inferiores se observó una reducción notable en este valor y para valores superiores a 5, aunque no aparece en la gráfica, se encontró que el consumo de tiempo fue exagerado o la memoria virtual desbordada.

Con el valor de AC fijo en 5 se ejecutó el algoritmo de recuperación estructural y semántica implementado por el entorno BeMantics y se obtuvo la Figura 5-9 la cual muestra el promedio de consumo de tiempo en la correspondencia en contraste con el promedio del número de nodos de los BP repositorio.

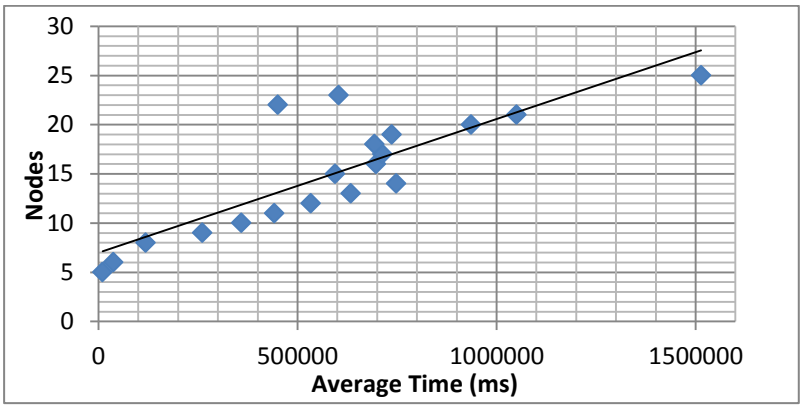


Figura 5-9 : Consumo de tiempo en la correspondencia vs el número de nodos de los BP.

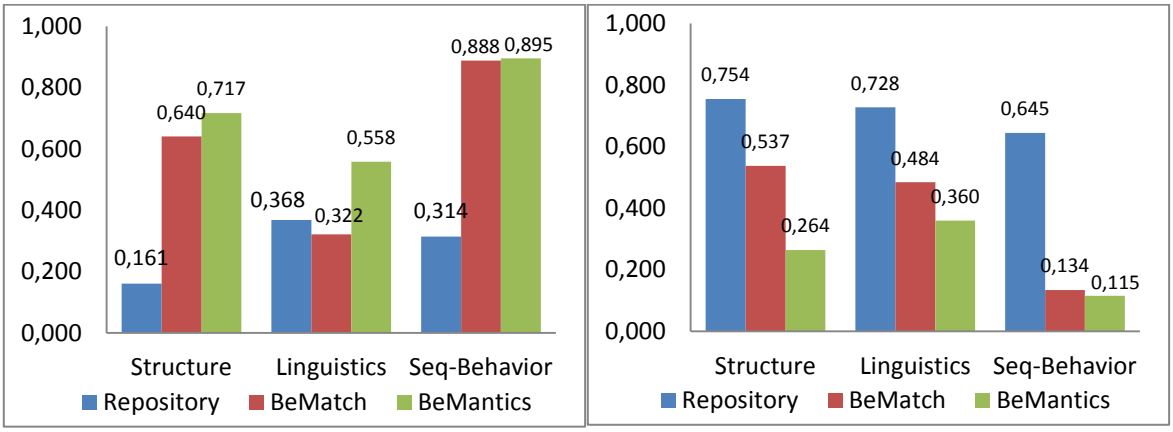
La Figura 5-9 muestra una línea de tendencia con un comportamiento lineal para el incremento del promedio de consumo de tiempo de acuerdo con el número total de nodos en cada BP repositorio. Por lo tanto, se puede afirmar que en general el consumo de tiempo para la correspondencia de dos BP con un número de nodos entre 5 – 30 puede presentar un comportamiento lineal de acuerdo con el incremento del número de nodos en cada grafo. Nótese además que el consumo de tiempo depende más de la selección de parámetros que del número de nodos en los BP.

5.5 Análisis de Relevancia

Este análisis se ejecutó para determinar las medidas gradadas promedio para tres mecanismos de recuperación de BP. El primero, es el mecanismo de indexación basado en semántica del comportamiento ejecutado por el repositorio el cual recupera BP utilizando los patrones de control de flujo; el segundo llamado BeMatch, el cual recupera BP utilizando criterios estructurales y léxicos; y el tercero es el mecanismo presentado en esta investigación denominado BeMantics, el cual utiliza un analizador estructural y semántico. En este punto cabe anotar que BeMantics corresponde a una versión mejorada de BeMatch, utilizando criterios semánticos en

lugar de criterios léxicos y además un repositorio que permite ejecutar un mecanismo de indexación (pre-correspondencia).

De esta manera, se compararon los valores de Pg y Rg entre los tres prototipos de acuerdo a criterios de estructura, lingüística y comportamiento secuencial, tal como lo muestra la Figura 5-10.



(a) (b)

Figura 5-10 : Valores promedio de Precisión (a) y Exhaustividad (b) gradadas para el repositorio, BeMantics y BeMatch.

Comparando la precisión gradada (Figura 5-10 (a)), BeMantics alcanzó los valores más altos (0.717, 0.558 y 0,895) en los tres criterios, seguido por BeMatch (0.640, 0.322 y 0,888) y el repositorio con los valores más bajos (0.161, 0.368, y 0.314). Esto implica que BeMantics fue el mecanismo, para recuperar BP, que demostró mayor precisión, es decir con valor mínimo de fasos positivos. Además, se puede observar que el criterio que mejor se valoró en BeMantics respecto a BeMatch fue el criterio Lingüístico en un factor de 23.6%, lo cual demuestra que el uso de inferencia semántica en los nodos de tipo función de los BP permitió reducir en un 23.6% el número de falsos positivos (es decir los BP que se recuperaron pero que no son relevantes).

Sin embargo, comparando los valores para la exhaustividad gradada (Figura 5-10 (b)), BeMantics obtuvo el menor valor (0.264, 0.360 y 0.115) en todos los criterios, seguido por BeMatch (0.537, 0.484 y 0.134) y el repositorio con los valores más altos (0.754, 0.728 y 0.645). Esto significa que BeMantics tuvo mayor

probabilidad de entregar falsos negativos (es decir BP relevantes que no se recuperaron); lo cual se pudo producir debido a que algunos procesos relevantes se perdieron en la fase de pre-correspondencia ejecutada por el repositorio, aun cuando este mostró valores altos de Rg.

Para finalizar, en los resultados, el repositorio alcanzó el mejor valor de exhaustividad Rg y el peor valor de precisión Pg. Esto es normal dado que el repositorio fue diseñado como un filtro inicial con un alto valor de Rg (valores bajos de falsos negativos) y en este caso el valor de Pg no es tan importante debido a que se supone que los algoritmos de correspondencia deben mejorar la precisión, tal como se pudo observar en la Figura 5-10 (b).

Para más extensión en este análisis, se puede encontrar el conjunto total de resultados en el Anexo I

5.6 Resumen

En este capítulo se presentaron los resultados de los análisis realizados para evaluar el rendimiento y la precisión del entorno BeMantics en comparación con otra herramienta denominada BeMatch. Los resultados demostraron que el prototipo más preciso fue BeMantics pero también fue el que tuvo el valor más bajo de exhaustividad. Por otra parte, se analizó la fase de recuperación implementada por el repositorio, la cual demostró un valor bajo de precisión, pero con un valor alto de exhaustividad; esto indica que el repositorio recupera procesos de negocio no relevantes, pero reduce el número de falsos negativos.

Capítulo 6

6 CONCLUSIONES

La búsqueda de procesos se ha convertido en un campo crítico en muchas áreas, debido a la dificultad de encontrar servicios que se adapten exactamente a las exigencias de los usuarios. Por esta razón se puede llegar a incurrir en un consumo innecesario de recursos destinados a la búsqueda del proceso adecuado. De esta manera, el desarrollo de nuevos métodos de búsqueda, intuitivos, dotados de inteligencia artificial, basados en semántica y que reconozca lo que realmente el usuario necesita encontrar, permite agilizar el despliegue y configuración de nuevos BP y se convierten en un área importante de la I+D.

El presente trabajo presentó a BeMantics, un entorno que direcciona la recuperación de BP a través de mecanismos correspondencia en las perspectivas de comportamiento, estructura y semántica. Además se propuso un método de pre-correspondencia denominado “Semántica del Comportamiento”, el cual consiste en un método de indexación basado en patrones de control de flujo y sus relaciones semánticas. De esta manera, el diseño de BeMantics se realizó utilizando dos módulos principales; primero un repositorio con mecanismos de pre-correspondencia basada en semántica del comportamiento y segundo un mecanismo de correspondencia estructural y semántica el cual refina los resultados del repositorio utilizando un algoritmo de corrección de errores. Este algoritmo compara nodo a nodo un proceso de consulta con cada uno de los procesos obtenidos del repositorio y ejecuta operaciones de edición sobre estos últimos procesos con el fin de hacerlos tan similares como sean posibles al proceso de consulta.

Para analizar los resultados del entorno BeMantics se generó una plataforma Web de evaluación de pertinencia la cual permitió que un grupo de jueces humanos emita juicios de relevancia sobre un conjunto de 100 BP. Estos juicios de relevancia generaron un sub-conjunto de los BP considerados como relevantes, en otras palabras, aquellos BP que se esperaría que una herramienta automática recupere de acuerdo a un BP de consulta. El conjunto de BP relevantes obtenidos por los jueces resultó útil para la evaluación de relevancia del entorno BeMantics., la cual se ejecutó comparando los resultados de BeMantics con un trabajo previo denominado BeMatch, el cual contiene un analizador estructural muy similar, pero que carece de los módulos de pre-correspondencia y de análisis semántico sobre la tareas de los BP.

La evaluación de relevancia demostró que la aplicación de la semántica sobre la comparación de las tareas de los procesos permitió incrementar la precisión gradada pero redujo la exhaustividad gradada de BeMantics respecto a BeMatch el cual solo ejecutó una comparación léxica. Por lo tanto se puede decir que BeMantics fue más preciso que BeMatch pero perdió algunos procesos relevantes debido a la fase de pre-correspondencia. Lo anterior permitió concluir que el uso de ontologías de un dominio tan específico, como las telecomunicaciones, limita la capacidad semántica que se podría obtener al utilizar ontologías de dominios más generales, ya que el conjunto de conceptos con los que se pueden etiquetar los nombres e interfaces de las tareas se reduce al conjunto de conceptos incluidos en la ontología de dominio.

Por lo tanto, la presente investigación recomienda el uso de ontologías de dominios más generales ya que ofrecen un grupo más abierto de conceptos, empero, si el contexto dónde se ejecutarán los BP está muy bien definido una ontología de dominio específico podría ayudar a reducir ambigüedades entre los profesionales que se encarguen del BP de etiquetado de los BP.

Por otra parte, se realizó una evaluación de rendimiento la cual demostró que el repositorio presentó un bajo consumo de tiempo (10-40ms) y consecuentemente si se aplica como una fase de pre-correspondencia permite reducir el espacio de

búsqueda y el consumo total de tiempo para el entorno BeMantics en general, el cual presentó un rendimiento muy pobre.

A continuación se hace un resumen de los principales resultados obtenidos en la ejecución del presente trabajo de grado:

6.1 Resultados

A continuación se presenta un resumen de los principales resultados obtenidos con la ejecución del presente trabajo de grado:

- **Arquitectura para un repositorio de modelos BP.** A partir del estudio de diferentes repositorios de modelos de procesos existentes en el contexto empresarial y académico, fue posible formular una arquitectura especialmente diseñada para el almacenamiento y recuperación de modelos de BP.

- **Sistema de Indexación de grafos de modelos de BP.** Entre los resultados del estudio anterior, también se encontró que varios de estos repositorios de modelos de procesos carecen de un método de indexación por lo que ésta contribución constituye un aporte significativo para este trabajo.

- **Mecanismo de búsqueda semántico de modelos de BP basado en patrones de control de flujo.** La adición del factor semántico en las búsquedas al interior del repositorio, permite dotar al sistema de búsqueda la capacidad de inferir sobre las instancias de conceptos pertenecientes a una ontología y calcular la distancia semántica entre procesos asociando patrones de control de flujo.

- **Prototipo de búsqueda y almacenamiento de modelos de BP basado en semántica del comportamiento.** Creación de una aplicación que permite el almacenamiento y la recuperación de modelos de BP soportado en el mecanismo de búsqueda semántico mencionado anteriormente.

- **Definición de una ontología de patrones de control de flujo.** Esta definición permite realizar un razonamiento semántico y la inferencia sobre las instancias de los conceptos de la ontología de patrones de control de flujo.

- **Ontología de patrones de control de flujo:** La generación de una ontología que describe las relaciones basadas en funcionalidad, comportamiento, especialización y composición, entre los 12 patrones básicos definidos en BPMO. Esta ontología fue formulada tomando como referencia las ontologías de los proyectos SUPER[8], SemBiz[9] y m3pe[10] y permite dotar de semántica al mecanismo de recuperación de Modelos BP.

- **Organizador Semántico de Resultados:** la definición de un mecanismo de comparación semántica de patrones de control de flujo entre BP descritos en BPMO para el repositorio. Este mecanismo se definió basado en el cálculo de la similitud semántica entre los parámetros que describen dichos patrones.

- **Estudio del estado actual del conocimiento sobre modelado de BP:** se realizó un estudio del estado actual del conocimiento en lenguajes y formalismos de modelado de BP

- **Estudio sobre enriquecimiento semántico de BP.** Se estudiaron y se adaptaron dos ontologías del dominio de las telecomunicaciones seTOM y sSID, y se realizó un tutorial sobre enriquecimiento semántico de identificadores e interfaces (entradas y salidas) de BP utilizando dichas ontologías.

- **Creación de un entorno de correspondencia estructural y semántica de BP:** se creó un entorno llamado BeMantics el cual permite descubrir BP desde perspectivas de semántica, estructura y comportamiento.

---

[8] Semantics utilised for Process Management within and between Enterprises (http://www.ip-super.org)
[9] SemBiz project (http://www.sembiz.org/)
[10] Multi Meta Model Process Engineering (http://m3pe.deri.ie/)

- **Plataforma de Evaluación de Pertinencia:** la cual permitió a un conjunto de jueces humanos emitir juicios de relevancia en un conjunto de BP, los cuales sirvieron de referencia para evaluar la efectividad del modelo BeMantics.

- **Banco de Procesos de Prueba:** se desarrolló un conjunto de 100 BP, de los cuales 60 pertenecen al dominio de las telecomunicaciones y 40 al dominio de geo-procesamiento.

- **Publicaciones y Conferencias:** A continuación se listan los artículos generados a partir del presente trabajo de grado.

    - Artículo “Plataforma para Evaluar Sistemas de Recuperación de BP” presentado en la revista de Investigaciones UCM la cual está indexada en categoría C del sistema publindex de Colciencias. - 2011

    - Artículo “Business Process Model Retrieval Based on Graph Indexing Method” publicado en el journal *Business Process Management Workshop* de Springer Berlin Heidelberg y aceptado también dentro del workshop “rBPM – reuse in Business Process Management” en el marco de la conferencia “Business Process Management” en Hoboken New Jersey, septiembre de 2010.

    - Artículo “Business Process Repository based on Control Flow Patterns” presentado en la quinta conferencia euroamericana de sistemas telemáticos e informáticos (EATIS), en Panamá – Septiembre de 2010.

    - Artículo “Comparación Semántica de Tareas entre BP de Telecomunicaciones” presentado en el 5 Seminario Nacional de Tecnologías Emergentes en Telecomunicaciones y Telemática, Popayán – Junio – 2010

    - Artículo “Business Process Retrieval based on Behavioral Semantics” presentado a la revista EIA. En la actualidad se encuentra en fase de evaluación.

- **Asesorías en Trabajos de Grado:** A continuación se listan los trabajos de grado en los cuales el autor del presente trabajo actuó en calidad de co-director.

    - Codirección en la Monografía de pregrado “BÚSQUEDA SEMÁNTICA EN UN REPOSITORIO DE PROCESOS DE NEGOCIO” realizada por los ingenieros David Corchuelo y Daniel Rivas

    - Codirección en la Monografía de pregrado “COMPARACIÓN SEMÁNTICA DE TAREAS ENTRE DOS PROCESOS DE NEGOCIO DE TELECOMUNICACIONES” realizada por los ingenieros Adriana Bastidas y Leandro Ordóñez.

    - Codirección en la Monografía de pregrado “PLATAFORMA PARA LA EVALUACIÓN DE SISTEMAS DE RECUPERACIÓN DE SERVICIOS BASADOS EN COMPORTAMIENTO” realizada por la ingeniero Laura Sandino.

- **Pasantía Investigativa:** Se realizó una pasantía investigativa en la Universidad Politécnica de Turín (Turín, Italia -2010).

- **Actividades de Docencia:** Informe de actividades de docencia ejecutadas entre el periodo I -2008 al I-2010.

- **Material Digital:** Un disco compacto que contiene toda la información en formato digital generada en el transcurso del proyecto. Incluida la presente monografía, los anexos, un video tutorial de enriquecimiento semántico de tareas de BP y el código fuente del prototipo que implementó el entorno BeMantics.

- **Anexos:** Documentos anexos que presentan información adicional útil de complemento de esta monografía.

6.2 Trabajo Futuro

A continuación se listan los trabajos futuros que se plantean como continuación del presente proyecto de investigación.

Incrementar los métodos de recuperación de BP

Como trabajo futuro se pretende realizar mecanismos de minería de BP, de tal manera que el sistema no solo pueda realizar descubrimientos en los cuatro niveles previamente mencionados, sino que además, aprenda de acuerdo a las preferencias de los usuarios y a los patrones de comportamiento que más se ejecuten dentro de los BP permitiendo realizar una búsqueda personalizada de servicios.

Implementación de una ontología que contenga las relaciones existentes entre todos los patrones de control de flujo de referencia viables en BPMO

La ontología implementada (bpmooner-1.0.0.wsml) contempla solamente 12 patrones básicos de BPMO. Este trabajo de grado se podría extender por medio de la implementación en el lenguaje WSML de una ontología que soporte la totalidad de los 21 patrones de control de flujo de referencia viables en BPMO.

Adicionar a la capa de transformación de BP el soporte para varios lenguajes de modelado adicionales

A partir de la arquitectura propuesta se propone la implementación de transformadores que permitan generar a partir de diferentes notaciones de modelado de BP, tales como BPMN, XPDL, BPEL, etc., una representación formal unificada basada en grafos.

Plataforma para la búsqueda y composición de BP semánticos

Es posible crear una plataforma que gestione de manera automática los modelos de procesos y que además de realizar búsqueda de modelos de BP, permita un entorno de composición de procesos con los resultados generados a partir del entorno BeMantics. El usuario podría utilizar la plataforma para visualizar los modelos de procesos y a partir de los resultados del Ranking, generar nuevos modelos de procesos.

Experimentación del entorno BeMantics en un entorno empresarial real

Se propone realizar la experimentación del prototipo BeMantics en un entorno empresarial real, para determinar su desempeño y validar su utilidad en actividades de gestión y reingeniería de procesos de una empresa.

Mejorar el rendimiento del mecanismo de correspondencia estructural.

Se propone utilizar técnicas basadas en heurísticas para mejorar el rendimiento en cuanto a tiempo de ejecución del analizador estructural y semántico dado su alto consumo de recursos y tiempo.

Mejorar la Plataforma de evaluación de pertinencia de la recuperación de BP

Se propone el desarrollo de un módulo de evaluación de efectividad que reciba directamente los resultados de las herramientas automáticas, y los compare con los valores de relevancia obtenidos, aplique las ecuaciones para medir la efectividad de la recuperación, y despliegue las gráficas que describan esta medición. Este nuevo módulo ahorraría tiempo en análisis de datos y facilitaría la valoración de la efectividad de un algoritmo.

7 Referencias

[1] E. Gonçalves da Silva, L. Ferreira Pires, and M. van Sinderen, "Towards runtime discovery, selection and composition of semantic services," *Computer Communications,* pp. 159-168, 2011 2011.

[2] M. Mongiello and D. Castelluccia, "Modelling and verification of BPEL business processes," *Model-Based Methodologies for Pervasive and Embedded Software, International Workshop on,* vol. 0, pp. 144-148, 2006.

[3] S. I. U. Tibco, "Extending the Benefits of SOA beyond the Enterprise," Tibco Software Inc. UK2006.

[4] C. Pedrinaci, J. Domingue, and A. Sheth, "Semantic Web Services," J. Domingue, D. Fensel, and J. Hendler, Eds., ed: Springer, 2010.

[5] Ministerio-de-Comunicaciones, "Plan Nacional de Tecnologías de la Información y las Comunicaciones 2008 - 2019," ed. Bogotá D.C: Ministerio de Comunicaciones, 2008.

[6] T. Pollet, G. Maas, J. Marien, and A. Wambecq, "Telecom Services Delivery in a SOA," in *AINA (2)*, ed: IEEE Computer Society, 2006, pp. 529-533.

[7] Revista-Portafolio. (2008, 06/11/2011). *Crece número de adquisiciones, fusiones, alianzas y otros tipos de integraciones de compañías*. Available: http://www.portafolio.com.co/negocios/empresas/2008-03-14/ARTICULO-WEB-NOTA_INTERIOR_PORTA-4011446.html

[8] O. f. E. C.-o. a. D. Inter-American Development Bank. (2005, Concentraciones empresariales 2000-225 Analizadas por la fiscalía Nacional Económica de Chile. *(Foro Latinoamericano de Compentencia)*. Available: http://biografias.bcn.cl/alegislativo/pdf/cat/docs/3618-03/532.pdf

[9] D. Ferreiro, "Integración Cooperativa," *Jornada Temática "Empresa Agraria y Cooperativismo" Madrid, 2002,* 2002.

[10] C. Guerra M, "Fusiones Bancarias: El Caso Bancomer," Universidad Autónoma de México, México, 2002.

[11] Hewlet-Packard. (2009). *Integración empresarial*. Available: http://www.hp.com/latam/ar/servicios/aplicaciones_empresariales/enter_empresarial.html

[12] M. Livanos C, "El papel "vital" de las empresas," in *BBC. MUNDO.com*, ed: BBC, 2003.

[13] N. Pérez C, H. Muñoz, D. d. F. Marcos, and J. Martínez E. (2008, 06/12/2011). Gestión de Procesos de Negocio Semánticos. Available: http://www.ip-super.org/res/Papers/SBPM-TelecomID2007.pdf

[14] E. Stroulia and Y. Wang, "Structural and Semantic Matching for Assessing Web-service Similarity.," *Int. J. Cooperative Inf. Syst.,* 2005.

[15] N. Kokash, W. van den Heuvel, and V. D'Andrea, "Leveraging Web Services Discovery with Customizable Hybrid Matching," in *Proc. of ICSOC*, ed, 2006, pp. 522-528.

[16] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic Matching of Web Services Capabilities," in *Proc. of ISWC*, ed, 2002.

[17] B. Benatallah, M. S. Hacid, C. Rey, and F. Toumani, "Semantic Reasoning for Web Services Discovery," in *Proc. of ESSW*, ed, 2003.

[18] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with OWLS-MX," presented at the Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, Hakodate, Japan, 2006.

[19] C. Okkyung and H. Sangyong, "Flexible Rule-Based Web Services System for Users," *Next Generation Web Services Practices, International Conference on,* vol. 0, pp. 1-4, 2008.

[20] L. Lin and I. B. Arpinar, "Discovery of Semantic Relations between Web Services," *Web Services, IEEE International Conference on,* vol. 0, pp. 357-364, 2006.

[21] J. C. Corrales, D. Grigori, M. Bouzeghoub, and A. Gater, "Ranking BPEL Processes for Service Discovery," *IEEE Transactions on Services Computing,* vol. 3, pp. 178-192, 2010.

[22] R. Eshuis and P. W. P. J. Grefen, "Structural Matching of BPEL Processes," in *ECOWS*, ed: IEEE Computer Society, 2007, pp. 171-180.

[23] A. Wombacher and C. Li, "Alternative Approaches for Workflow Similarity," *2010 IEEE International Conference on Services Computing,* pp. 337-345, 2010.

[24] M. Fronk and J. Lemcke, "Expressing Semantic Web Service Behavior using Description Logics," *European Semantic Web Conference (ESWC). 2006.,* 2006.

[25] J. Hidders, M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede, and J. Verelst, "When are two Workflows the Same?," in *CATS* vol. 41, ed: Australian Computer Society, 2005, pp. 3-11.

[26] I. Markovic, "Enhanced Process Query Framework," Germany Patent 09001690.8, 2009.

[27] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens, "Process discovery in event logs: An application in the telecom industry," *Applied Soft Computing,* vol. 11, pp. 1697-1710, 2011.

[28] A. J. M. M. Weijters, W. M. P. Van Der Aalst, and A. K. Alves De Medeiros, "Process Mining with the HeuristicsMiner Algorithm," *Technology,* vol. 166, pp. 1-34, 2006.

[29] W. Van Der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, pp. 1128-1142, 2004.

[30] M. Sellami, S. Tata, and B. Defude, "Service Discovery in Ubiquitous Environments: Approaches and Requirements for Context-Awareness," in *Business Process Management Workshops* vol. 17, ed: Springer, 2008, pp. 516-522.

[31] R. Nayak and B. Lee, "Web Service Discovery with additional Semantics and Clustering," presented at the Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2007.

[32] B. Sapkota, "Web Service Discovery in Distributed and Heterogeneous Environment," *WWW Service Composition with Semantic Web Services (wscomps05) Workshop, Compiegne, France,* September, 2005. 2005.

[33] J. C. Corrales, "Behavioral matchmaking for service retrieval.," PhD Computer Science, Computer Science, University of Versailles Saint-Quentin-en-Yvelines, Versailles, 2008.

[34] G. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM,* vol. 38, pp. 39-41, 1995.

[35] CRM-Management-Editors. (2009, 06/12/2011). *Philippine telco cuts time to market for new services by 80%*. Available: http://www.enterpriseinnovation.net/content/philippine-telco-cuts-time-market-new-services-80

[36] D. Ramirez and J. Rojas, "Lineamientos para Composición de Servicios de Telecomunicaciones en un Entorno JAIN SLEE basados en Software de Libre Distribución," Ingeniería Electrónica y Telecomunicaciones B.Tech, Departamento de Ingeniería Telemática, Universidad del Cauca, Popayán, 2010.

[37] OASIS, "Reference Architecture Foundation for Service Oriented Architecture Version 1.0," vol. 2011, ed: OASIS, 2011.

[38] K. Chu, O. Cordero, M. Korf, C. Pickersgill, and R. Whitmore, "Introduction to SOA and the Oracle SOA Suite," in *Oracle SOA Suite Developer's Guide 10g (10.1.3.1.0)*, Oracle, Ed., ed Redwood City, CA: Oracle, 2006, pp. 1.1 - 1.10.

[39] The-Open-Group. (2011, 18-Nov). *The SOA Work Group : Definition of SOA*. Available: http://www.opengroup.org/soa/soa/def.htm

[40] IBM. (2008). *Service Oriented Architecture — SOA*. Available: http://www-01.ibm.com/software/solutions/soa/

[41] M. A. C. Bhakti and A. B. Abdullah, "Formal Modelling of an Autonomic Service Oriented Architecture " presented at the ICCCM2011, Sydeny, Australia., 2011.

[42] S. Otón, "Propuesta de una arquitectura software basada en servicios para la implementación de repositorios de objetos de aprendizaje distribuidos," PhD PhD, Departamento de Ciencias de la Computación, Universidad de Alcalá, Alcalá, España, 2006.

[43] W3C, "RDF/XML Syntax Specification," W3C2004.

[44] J. Hendler and D. McGuiness, "The DARPA Agent Markup Language," *IEEE Intelligent Systems,* pp. 6-7, 2000.

[45] D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein, "OIL in a Nutshell," in *Proc1 of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKA1 2000)*, ed: Springer, 2000, pp. 1-16.
[46] W3C, "OWL Web Ontology Language Guide," 2004.
[47] W3C. (2010, 12-01-2001). *Guía Breve de Servicios Web*. Available: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb
[48] W3C, "Web Services Description Language (WSDL) 1.1," in *W3C Note*, ed: W3C, 2001.
[49] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," presented at the Proceedings of the 22nd international conference on Software engineering, Limerick, Ireland, 2000.
[50] M. Mongiello and D. Castelluccia, "Modelling and Verification of BPEL Business processes," IEEE Computer Society, Bari Italia., 2006
[51] The-WfMC, "Workflow Management Coalition Terminology & Glossary," The-WfMC, Winchester-UK1999.
[52] S. A. White, "Introduction to BPMN," ed: IBM Software Group, 2006.
[53] W. M. P. V. D. Aalst, A. H. M. T. Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns," *Distrib. Parallel Databases,* vol. 14, pp. 5-51, 2003.
[54] N. Russell, A. H.M, t. Hofstede, W. M. P. v. d. Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View. ," BPM Center *BPM-06-22*, 2006.
[55] R. K. L. Ko, S. S. G. Lee, and E. W. Lee, "Business process management (BPM) standards: a survey," *Business Process Management Journal,* vol. 15, pp. 744-791, 2009.
[56] W. M. P. Van Der Aalst, A. H. M. Hofstede, and M. Weske, "Business Process Management: A Survey," in *1st International Conference on Business Process Management*, Eindhoven, The Netherlands, 2003, pp. 1-12.
[57] G. Keller, M. Nuttgens, and A. W. Scheer. (1992, Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK).
[58] W. M. P. v. d. Aalst, "Formalization and Verification of Event-driven Process Chains," ed. Eindhoven University of Technology: Department of Mathematics and Computing Science, 2000.
[59] S. A. White, "Introduction to BPMN," 2004.
[60] J.-J. Dubray. (2008, 20/11/2011). *BPML*. Available: http://www.ebpml.org/bpml.htm
[61] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guízar, N. Kartha, C. K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. v. d. Rijn, P. Yendluri, and A. Yiu, "Web Services Business Process Execution Language Version 2.0," ed: OASIS Standard, 2007.
[62] J. C. Corrales, D. Grigori, and M. Bouzeghoub, "BPEL Processes Matchmaking for Service Discovery," Universite de Versailles Saint-Quentin, 2005.
[63] A. E. Botaro, F. P. Lozano, and I. M. Bulo, "Operadores de mutación para WS-BPEL 2.0," 2008.

[64] WfMC. (2011, 21/11/2011). *XPDL Support and Resources*. Available: http://www.wfmc.org/xpdl.html
[65] D. Hollingsworth. (1995, The Workflow Reference Model Version 1.1.
[66] A. Gater, D. Grigori, and M. Bouzeghoub, "Matching and similarity evaluation of OWL-S process models," Universite de Versailles Saint-Quentin, 2008.
[67] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. (2004, Febrero 12 de 2010). *OWL-S: Semantic Markup for Web Services*. Available: http://www.w3.org/Submission/OWL-S/
[68] Z. Yan, E. Cimpian, M. Mazzara, and M. Zaremba, "D1.1 BPMO Requirements Analysis and Design," SemBiz02/2007 2007.
[69] P. E. Black. (2008, 22/11/2011). *Finite state machine*. Available: http://xlinux.nist.gov/dads//HTML/finiteStateMachine.html
[70] J. Daciuk, B. W. Watson, S. Mihov, and R. E. Watson, "Incremental construction of minimal acyclic finite-state automata," *Comput. Linguist.,* vol. 26, pp. 3-16, 2000.
[71] M. Havey, "The Pi-calculus," in *Essential Business Process Modeling* ed: O'Reilly Media, 2005, p. 122.
[72] B. Hendrickson and R. Leland, "A multilevel algorithm for partitioning graphs," presented at the Proceedings of the 1995 ACM/IEEE conference on Supercomputing, San Diego, California, United States, 1995.
[73] K. Xu, L. Liu, and C. Wu, "A three-layered method for business processes discovery and its application in manufacturing industry," *Computers in Industry,* vol. 58, pp. 265-278, 2007.
[74] A. Koschmider, T. Hornung, and A. Oberweis, "Recommendation-based editor for business process modeling," *Data Knowl. Eng.,* vol. 70, pp. 483-503, 2011.
[75] P. Châtel, "Toward a Semantic Web Service Discovery and Dynamic Orchestation based on the Formals specification of Functional Domain Knowledge," *ICSSEA 2007,* 2007.
[76] J. Kopecky, T. Vitvar, C. Bournez, and J. Farrell, "SAWSDL: Semantic Annotations for WSDL and XML Schema," *IEEE Internet Computing,* vol. 11, pp. 60-67, 2007.
[77] C. Kiefer, A. Bernstein, H. J. Lee, M. Klein, and M. Stocker, "Semantic Process Retrieval with iSPARQL," presented at the Proceedings of the 4th European conference on The Semantic Web: Research and Applications, Innsbruck, Austria, 2007.
[78] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, and J. Lee, "On Accommodating Inter Service Dependencies in Web Process Flow Composition," in *In AAAI Spring Symposium on SWS*, ed, 2004, pp. 37-43.
[79] S. Sakr and A. Awad, "A framework for querying graph-based business process models," *Proceedings of the 19th international conference on World wide web WWW 10,* pp. 1297-1297, 2010.
[80] A. Wombacher, B. Mahleko, P. Fankhauser, and E. Neuhold, "Matchmaking for Business Processes based on Choreographies," in *Proc. of EEE*, ed, 2004.
[81] B. Mahleko and A. Wombacher, "Indexing Business Processes based on Annotated Finite State Automata," in *Proc. of The ICWS*, ed, 2006.

[82] Z. Shen and J. Su, "Web Services Discovery Based on Behavior Signatures," in *Proc. of IEEE SCC*, ed, 2005.

[83] B. Yun, J. Yan, M. Liu, and Y. Yu, "Behavioral Equivalence Based Web Service Discovery," presented at the Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02, 2008.

[84] A. K. A. D. Medeiros, B. F. V. Dongen, W. M. P. V. D. Aalst, and A. J. M. M. Weijters, "Process Mining : Extending the α-algorithm to Mine Short Loops," *Technology,* 2004.

[85] A. K. A. Medeiros and A. J. M. M. Weijters, "Genetic process mining: an experimental evaluation," *Data Mining and Knowledge Discovery,* vol. 14, pp. 245-304, 2005.

[86] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust process discovery with artificial negative events," *Journal of Machine Learning Research,* vol. 10, pp. 1305-1340, 2009.

[87] K. Rosen, "Isomorphism of Graphs," in *Discrete Mathematics and Its Applications*. vol. 6, M.-H. Science, Ed., 6 ed: McGraw-Hill Science/Engineering/Math; 6 edition, 2006, pp. 560-563.

[88] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," *Order A Journal On The Theory Of Ordered Sets And Its Applications,* vol. 02, pp. 721-724, 2002.

[89] X. Yan and J. Han, "CloseGraph: mining closed frequent graph patterns," 2003, pp. 286-295.

[90] X. Y. F. Zhu, J. Han, and P. S. Yu, "gPrune: A Constraint Pushing Framework for Graph Pattern Mining," *PAKDD'07,* 2007.

[91] R. Giugno and D. Shasha, "GraphGrep: A Fast and Universal Method for Querying Graphs.," in *Proc. of ICPR*, ed, 2002, pp. 112-115.

[92] A. Ferro, R. Giugno, M. Mongioví, A. Pulvirenti, D. Skripin, and D. Shasha, "Graphblast: Multi-Feature Graphs Database Searching," presented at the Workshop on Network Tools and Applications in Biology (NETTAB), Pisa, 2007.

[93] R. G. A. Ferro, M. Mongiovi', A. Pulvirenti, D. Skripin, D. Shasha . "Graphblast: Multi-Feature Graphs Database Searching," presented at the Workshop On Network Tools And Applications In Biology - NETTAB 2007., 2007.

[94] S. Srinivasa and M. H. Singh, "GRACE: A Graph Database System," in *International Conference on Management of Data*, Hyderabad, India, 2005.

[95] X. Yan, P. S. Yu, and J. Han, "Graph indexing: a frequent structure-based approach," presented at the Proceedings of the 2004 ACM SIGMOD international conference on Management of data, Paris, France, 2004.

[96] J. W. T. Jin, N. Wu, M. La Rosa and A.H.M. ter Hofstede, "Efficient and Accurate Retrieval of Business Process Models through Indexing. ," Queensland University of Technology 2010.

[97] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *J. ACM,* vol. 23, pp. 31-42, 1976.

[98] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 26, pp. 1367-1372, 2004.

[99] Oracle. (2010, Nov. 16th). *Oracle Berkeley DB Java Edition*. Available: http://www.oracle.com/technetwork/database/berkeleydb/documentation/index-160410.html

[100] M. Dimitrov, V. Momtchev, A. Simov, D. Ognyanoff, and M. Konstantinov. (2006, Nov. 24th). *wsmo4j Programmers Guide v. 2.0.1* [Programmer Guide]. Available: http://wsmo4j.sourceforge.net/doc/wsmo4j-prog-guide.pdf

[101] J. C. Corrales, D. Grigori, and M. Bouzeghoub, "BPEL Processes Matchmaking for Service Discovery.," in *Proc. of CoopIS*, ed, 2006.

[102] R. Giugno and D. Shasha, "GraphGrep: A fast and universal method for querying graphs," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2002, pp. 112-115 vol.2.

[103] F. Zhu, X. Yan, J. Han, and P. Yu, "gPrune: A Constraint Pushing Framework for Graph Pattern Mining," in *Advances in Knowledge Discovery and Data Mining*. vol. 4426, Z.-H. Zhou, H. Li, and Q. Yang, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 388-400.

[104] J. Edwards, "Oracle Berkeley DB Java Edition," 2006.

[105] J. Ge and Y. Qiu, "Concept Similarity Matching Based on Semantic Distance," presented at the Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid, 2008.

[106] B. Messmer, "Graph Matching Algorithms and Applications," University of Bern, 1995.

[107] A. Patil, S. Oundhakar, A. Sheth, and K. Verna, "METEOR-S Web Service Annotation Framework," in *13th International Conference on the World Wide Web*, 2004.

[108] Z. Shangguan, Z. Gao, and K. Zhu, "Ontology-Based Process Modeling Using eTOM and ITIL," in *CONFENIS (2)* vol. 255, ed: Springer, 2007, pp. 1001-1010.

[109] *Enhanced Telecom Operations Map (eTOM),* I. T. Map Recommendation, 2005.

[110] TM-Forum. (2005). *Information Framework (SID) in Depth*. Available: http://www.tmforum.org/InformationFramework/6647/home.html

[111] J. Martínez and N. Pérez, "YATOSP: Marco de Referencia Semántico para el Sector Telco," presented at the Telecom I+D Bilbao 08, Bilbao, Spain, 2008.

[112] S. Grimm, U. Keller, H. Lausen, and G. Nagypál, "A Reasoning Framework for Rule-Based WSML," in *ESWC* vol. 4519, ed: Springer, 2007, pp. 114-128.

[113] J. C. Corrales, D. Grigori, M. Bouzeghoub, and J. E. Burbano, "BeMatch: A Platform for Matchmaking Service Behavior Models," in *11th International Conference on Extending Database Technology, EDBT*, ed, 2008.

[114] C. Figueroa, L. Sandino, and J. C. Corrales. (2011, 05/2011) Plataforma para Evaluar Sistemas de Recuperación de Procesos de Negocio. *Revista de Investigaciones UCM* [Research Paper]. 64-76. Available: http://hdl.handle.net/10839/210

[115] A. Bernstein, E. Kaufmann, C. Bürki, and M. Klein, "How Similar Is It? Towards Personalized Similarity Measures in Ontologies," in *7th International Conference Wirtschaftsinformatik*, ed Bamberg, Germany: Physica-Verlag HD, 2005, pp. 1347-1366.

[116] A. Goderis, P. Fisher, A. Gibson, F. Tanoh, K. Wolstencroft, D. De Roure, and C. Goble, "Benchmarking workflow discovery: a case study from bioinformatics," *Concurrency and Computation: Practice and Experience,* vol. 21, pp. 2052-2069, 2009.

[117] D. C. Blair, *Language and representation in information retrieval*: Elsevier North-Holland, Inc., 1990.

[118] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc., 1999.

[119] P. Borlund, "Experimental components for the evaluation of interactive information retrieval systems," *Journal of Documentation,* vol. 56, pp. 71-90, 2000.

[120] N. O. Pors, "Information retrieval, experimental models and statistical analysis," *Journal of Documentation,* vol. 56, pp. 55 - 70, 2000.

[121] U. Küster and B. König-Ries, "On the Empirical Evaluation of Semantic Web Service Approaches: Towards Common SWS Test Collections," presented at the Proceedings of the 2008 IEEE International Conference on Semantic Computing, 2008.