Universidad del Cauca
Grupo de Ingeniería Telemática

Recuperación Multinivel de Procesos de Negocio basado en Semántica del Comportamiento

Cristhian Figueroa
Juan Carlos Corrales
Gustavo Adolfo Ramirez

RECUPERACIÓN MULTINIVEL DE PROCESOS DE NEGOCIO BASADA EN SEMÁNTICA DEL COMPORTAMIENTO

Cristhian Figueroa

Juan Carlos Corrales

Gustavo Ramirez-Gonzalez

Grupo de Ingeniería Telemática

Universidad del Cauca

RECUPERACIÓN MULTINIVEL DE PROCESOS DE NEGOCIO BASADA EN SEMÁNTICA DEL COMPORTAMIENTO

ISBN-10 0-9833210-6-X
ISBN-13 978-0-9833210-6-4

Printing History:
April 2012. First Edition.
This section has been edited from printed version due to reserved space from the publisher

Information on this title: http://www.researchandinnovationbook.com/9780983321064.html

Cover Design: Lizeth Fernanda Jiménez R (lizethfjr@gmail.com)

Al apoyo de nuestras familias que motivaron el esfuerzo incansable y la persistencia infinita que nos hace cada día más grandes

Biografía

Cristhian Figueroa, es Ingeniero en Electrónica y Telecomunicaciones (2008), y Magister en Ingeniería Telemática (2012) de la Universidad del Cauca (Colombia). Actualmente es estudiante de Doctorado en Ingeniería Telemática en la Universidad del Cauca y la Universidad Politécnica de Turín (Italia). Además, es investigador del Grupo en Ingeniería Telemática de la Universidad del Cauca. Su interés de investigación está centrado en el desarrollo de nuevas técnicas para el descubrimiento de servicios, basadas en comportamiento lingüística y estructura, aplicadas en entornos web y móviles.

Juan Carlos Corrales, es Ingeniero en Electrónica y Telecomunicaciones (1999), y Magister en Ingeniería Telemática (2006) de la Universidad del Cauca. Recibió su título de Doctor en Ciencias Especialidad Informática (2008) en la Universidad de Versailles Saint-Quentin-en-Yvelines (Francia). Actualmente, es Profesor Titular y lidera el Grupo de Investigación en Ingeniería Telemática en la Universidad del Cauca. Su interés de investigación está centrado en el desarrollo de nuevas técnicas de recuperación de servicios basados en comportamiento y modelos semánticos; y en la composición automática en plataformas para despliegue de servicios.

Gustavo Ramirez-Gonzalez, es ingeniero en Electronica y Telecomunicaciones (2001) y Magister en Ingeniería Telemática (2006) de la Universidad del Cauca. Recibió su titulo de Doctor en Ingeniería Telemática (2010) en la Universidad Carlos III de Madrid (España). Actualmente es profesor titular e investigador del Departamento de Telemática en la Universidad del Cauca. Ha participado en proyectos nacionales e internacionales en Colombia y España. Sus áreas de interes son la computación móvil y ubicua, y los servicios avanzados de telecomunicaciones.

Agradecimientos

Los profesores Dr. Juan Carlos Corrales, Dr. Gustavo Ramírez y el candidato a Doctor Cristhian Figueroa, agradecemos a la Universidad del Cauca, el Grupo de Investigación en Ingeniería Telemática y el Departamento Administrativo de Ciencia y Tecnología (Colciencias) por el soporte financiero y científico; y por promover una comunidad investigativa que propende por la difusión del conocimiento en aras de una mejor sociedad y la reducción de la brecha tecnológica de nuestro país.

Agradecemos, además a la Universidad Politécnica de Turín, por la asistencia científica y académica; y a nuestras familias por el apoyo incondicional que mantiene nuestro ánimo encendido durante cada día de nuestras vidas.

Declaración de financiación

El presente trabajo ha sido financiado gracias a las siguientes fuentes:

- Universidad del Cauca (Colombia). Comisión de estudios: resolución VRA 0134 de 2010.
- Vicerrectoría de Investigaciones - Universidad del Cauca (Colombia). IV Convocatoria programa de apoyo a proyectos de investigación, desarrollo e innovación en el marco de maestrías y doctorados - Proyecto ID 2737.
- Departamento Administrativo de Ciencia y Tecnología - Colciencias (Colombia) y Universidad del Cauca. Programa Jóvenes Investigadores e Innovadores año 2008.
- Proyecto EM-COOPEN - Erasmus Mundus External Cooperation Window: solicitud CP10M0995CA.
- Departamento de Telemática y programa de Maestría en Ingeniería Telemática de la Universidad del Cauca, por la financiación parcial del desarrollo de este libro.
- Universidad del Cauca con la financiacón en labor académica del Proyecto “Descubrimiento Automático de Procesos de Negocio basado en Semántica del Comportamiento”.

Resumen

La recuperación de procesos de negocio (BP) es un paso fundamental para el reúso de componentes software dentro de nuevos productos y servicios informáticos. Sin embargo, este paso es crítico en muchas áreas y puede llegar a incurrir en un consumo innecesario de recursos debido a la dificultad de encontrar servicios que se adapten exactamente a las exigencias de los usuarios. Por consiguiente el desarrollo de métodos de búsqueda, intuitivos, dotados de inteligencia artificial, basados en semántica y que reconozcan lo que realmente el usuario necesita, se convierte en un área importante de la I+D debido a que permite agilizar el despliegue y la configuración de nuevos BP.

En este contexto, el presente libro resume los resultados de un proyecto de investigación científica adelantado por el Grupo de Ingeniería Telemática de la Universidad del Cauca (Colombia) en el cual se definió un entorno para la recuperación de BP denominado BeMantics. Este entorno fue desarrollado a través de dos módulos principales; un repositorio con mecanismos de pre-correspondencia basados en semántica del comportamiento; y un módulo de correspondencia estructural y semántica que utiliza un algoritmo de corrección de errores con el fin de refinar los resultados obtenidos del repositorio. Como resultado, el entorno BeMantics entrega una lista de BP ordenados de acuerdo a su similitud estructural, semántica y de comportamiento respecto a un BP de consulta. Este resultado es analizado posteriormente a partir de una plataforma Web de evaluación de pertinencia la cual facilita que un grupo de jueces humanos emita juicios de relevancia sobre un conjunto de BP de prueba. Estos juicios de relevancia permiten obtener un sub-conjunto de los BP considerados como relevantes, los cuales sirvieron como base para estudiar la relevancia de los resultados alcanzados con el entorno BeMantics.

Abstract

Retrieving business process (BP) is a fundamental step for reusing software components within new information services and software products. Nevertheless, this step is critical in many fields and it may incur in unnecessary resources consumption due to the complexity to find services matching exactly the user's requirements. Therefore, the development of intuitive, endowed with artificial intelligence and semantics-based methods which can recognize what the user really needs, become an important R&D area because it speeds up the deployment and configuration of new BP.

In this context, this book summarizes the results of a scientific research project conducted by Telematics Engineering Group of the University of Cauca (Colombia), in which a BP retrieval environment called BeMantics is defined. This environment is developed through two main modules: a repository module with pre-matching mechanisms based on behavioral semantics; and a structural, semantics and behavioral matching module which uses and error-correcting algorithm to refine the results obtained from repository. As a result the BeMantics environment provides a BP ranking according to their structural, semantics and behavioral similarity with regard to a query BP. This result is later analyzed through a pertinence evaluation web tool which enables a group of human judges to emit relevance judgments on a set of test BPs. These relevance judgments allow obtaining a subset of BP considered as relevant, which let the pertinence tool to be used as basis for studying the relevance of the results achieved with the BeMantics environment.

Tabla de Contenido

Listado de Figuras

Listado de Tablas

Capítulo 1

Introducción

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 (Gonçalves da Silva, Ferreira Pires, y van Sinderen, 2011).

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 el propósito de reunir sus funcionalidades individuales y lograr un objetivo de negocio común (Mongiello y Castelluccia, 2006).

En este sentido, SOA ha ganado una gran aceptación al interior de las compañías IT y la comunidad de investigación las cuales han promovido la proliferación de lenguajes, herramientas y componentes software reutilizables (Pedrinaci, Domingue, y Sheth, 2011; Tibco, 2006). 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 debe cumplir características de fácil reúso y bajo tiempo de salida al mercado (time-to-market) considerando, además, los requisitos de los usuarios.

1.1. 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 con el fin de satisfacer las necesidades de los clientes modernos (Mincomunicaciones-Colombia, 2008).

En esta situación pueden encontrarse 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 objeto de desplegar rápida y eficientemente nuevos servicios de valor agregado, adoptando un paradigma orientado al servicio que les proporcione la racionalización de su infraestructura IT (Pollet, Maas, Marien, y Wambecq, 2006). Sin embargo, lograr este objetivo no es fácil ya que 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 puede presentarse esta situación, es el caso de la integración horizontal o vertical de organizaciones, la cual es el 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 (Revista-Portafolio, 2008; Inter-American Development Bank y Oranisation for Economic Co-operation and Development, 2005; Ferreiro, 2002), 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 (Guerra M, 2002). Este fenómeno resulta fundamental en la región latinoamericana ya que permite fortalecer las empresas para que puedan afrontar con garantías el futuro tan cambiante del mercado (Hewlet-Packard, 2009).

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 administra sus BP de diferente manera. Un ejemplo de esto son las grandes compañías que utilizan un gran número de aplicaciones empresariales independientes en las cuales hacer un cambio requeriría del análisis y re-codificación de miles de interfaces (Livanos-Cattaui, 2003). Lo anterior pone en evidencia la necesidad de reorganizar los procesos internos de cada organización para 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 en la integración organizacional, sin embargo queda establecida la necesidad de modelos que faciliten la búsqueda de procesos existentes suficientemente similares al BP solicitado por el cliente, de tal manera que la integración sea rápida y tenga un menor impacto sobre los procesos que normalmente son ejecutados dentro de las organizaciones. Un ejemplo de empresas que padece este problema puede encontrarse 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, exige constantes ajustes en sus modelos de negocio. Además los clientes actuales ya no se conforman con los simples servicios de voz o datos; por el contrario, demandan servicios más avanzados e integrados (Pérez C, Muñoz, Marcos, y Martínez E, 2008). Para esto dichas empresas deben retomar los servicios existentes, complementarlos y en algunos casos integrarlos para crear nuevos BP más avanzados y adaptables a las necesidades de sus clientes.

1.2. Definición del Problema

En la actualidad existe una variedad de métodos de recuperación de componentes software, los cuales pueden clasificarse 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 interfaces (entradas y salidas) y los nombres de los servicios o tareas de los componentes software (Stroulia y Wang, 2005; Kokash, Heuvel, y DÁndrea, 2006). 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 (Gonçalves da Silva y cols., 2011; Paolucci, Kawamura, Payne, y Sycara, 2002; Benatallah, Hacid, Rey, y Toumani, 2003; Klusch, Fries, y Sycara, 2006; Okkyung y Sangyong, 2008; Lin y Arpinar, 2006). El nivel estructural compara componentes software estructurados (composiciones de WS representadas con BP) a través de algoritmos de isomorfismo de grafos (Corrales, Grigori, Bouzeghoub, y Gater, 2010; Eshuis y Grefen, 2007; Wombacher y Li, 2010). El nivel de comportamiento, al igual que el anterior, compara componentes software estructurados, pero en este caso utiliza criterios basados en el flujo de control (es decir constructores específicos que definen como se comporta un BP) (Fronk y Lemcke, 2006; Hidders, Dumas, van der Aalst, Hofstede, y Verelst, 2005; Markovic, 2009), o criterios basados en los registros de ejecución de los BP (Goedertier, De Weerdt, Martens, Vanthienen, y Baesens, 2011; Weijters, van der Aalst, y de Medeiros, 2006; van der Aalst, Weijters, y Maruster, 2004).

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, 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 adaptados a las exigencias de los usuarios (Sellami, Tata, y Defude, 2008; Nayak y Lee, 2007; Sapkota, 2005); además del uso de técnicas de indexación que permitan acelerar el proceso de descubrimiento.

Finalmente, de acuerdo con todas las consideraciones descritas en esta sección, se plantea la siguiente pregunta de investigación: ¿Cómo reducir los tiempos de despliegue de BP considerando una fase de descubrimiento adaptada a los requerimientos de búsqueda en cuanto a semántica del comportamiento?

1.3. Antecedentes

El estudio descrito en este libro parte de las bases cimentadas por el trabajo doctoral del Dr. Corrales (Corrales, 2008), en el cual se plantea el descubrimiento de BP teniendo en cuenta la 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 (Miller, 1995).

En este libro se presenta un estudio científico que complementa el trabajo del Dr. Corrales, a través de dos fases de descubrimiento: la primera, denominada fase de pre-correspondencia crea un índice basado en la "*semántica del comportamiento*", la cual no solo tiene en cuenta aspectos estructurales sino que presta especial atención al comportamiento en cuanto a la semántica del flujo de ejecución (flujo de control); y la segunda, denominada fase de correspondencia, realiza funcionalidades similares al trabajo del Dr. Corrales en cuanto a la correspondencia estructural de los BP, pero la comparación léxica de tareas es complementada con un enfoque semántico en un dominio específico de aplicación en el contexto de las telecomunicaciones.

1.4. Alcance

Este libro describe los resultados de un estudio científico realizado al interior del Grupo de Ingeniería Telemática de la Universidad del Cauca, alrededor las de técnicas de descubrimiento en los cuatro niveles descritos en la sección 1.2. A partir de estos resultados propone un mecanismo de recuperación divido en dos fases; la primera denominada de pre-correspondencia basada en semántica del comportamiento, es decir, semántica aplicada al flujo de control; y la segunda denominada correspondencia, la cual es una técnica multinivel que básicamente integra los niveles de interfaces, semántica y estructura.

Para este fin, se estudiaron y diseñaron 100 BP de los dominios de las telecomunicaciones y el 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 enriquecidos semánticamente en los tipos de datos de sus interfaces (entradas y salidas) y los nombres de sus tareas. En cuanto al tiempo de despliegue de servicios fueron tomados como referencia los trabajos de Nokia Siemens Networks e IBM (CRM-Management-Editors, 2009), los cuales demuestran 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 (Ramirez y Rojas, 2010) el cual 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 BP; puede afirmarse que los mecanismos de descubrimiento automáticos permiten acelerar el despliegue y configuración de nuevos BP y servicios a partir de la reutilización de los componentes recuperados.

1.5. Resumen

En este capítulo fueron presentados los escenarios de motivación del estudio científico desarrollado por el Grupo de Ingeniería Telemática de la Universidad del Cauca al rededor del tema de descubrimiento de BP. Este capítulo define, además, el problema central y los antecedentes desde los cuales parte la investigación.

Capítulo 2

Estado actual del conocimiento

Este capítulo describe el estado actual del conocimiento alrededor de la recuperación de BP. Inicialmente la sección 2.1 presenta la base conceptual sobre la cual se fundamenta la investigación descrita en este libro y a continuación la sección 2.2 expone 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 (OASIS, 2011). 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 (Gonçalves da Silva y cols., 2011). Por esta razón, se ha convertido en una tendencia fundamental para el diseño, desarrollo e integración de nuevos productos software a través de la orientación al servicio (Chu, Cordero, Korf, Pickersgill, y Whitmore, 2006; The-Open-Group, 2011) integrando negocios mediante tareas de negocio representadas por componentes software modulares, reutilizables y ampliamente escalables.

Dichos componentes software son conocidos como servicios complejos y pueden estar constituidos por otros servicios más simples con el propósito de reunir sus funcionalidades y cumplir un fin de negocio común (IBM, 2008; Bhakti y Abdullah, 2011).

2.1.2. 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 (The-Open-Group, 2011). Además, pueden estar descritos, publicados, localizados e invocados en la red a través de estándares abiertos de internet que facilitan la interoperabilidad entre distintas aplicaciones (W3C, 2001a).

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)(W3C, 2001b), el cual permite describir servicios en un formato XML(Extensible Markup Language) y facilita el intercambio de mensajes entre servicios a través del protocolo SOAP (Simple Object Access Protocol). El segundo, conocido como RESTFul es una técnica para implementar WS utilizando el protocolo HTTP y los principios del estilo arquitectónico REST (Representational State Transfer) inicialmente definido por Fielding y Taylor (Fielding y Taylor, 2000).

2.1.3. Ontología

Ontología es un concepto tomado de la filosofía y aplicado a la informática, que facilita inferir conceptos sobre un dominio del conocimiento. En la informática, las ontologías se han convertido en una de las herramientas más importantes de las web Semántica, ya que de acuerdo con el W3C (World Wide Web Consortium) permiten "definir formalmente un conjunto común de términos que se utilizan para describir y representar un dominio" (Kokash y cols., 2006). En este sentido el uso de ontologías dota a los sistemas de búsqueda de información con elementos 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 actividades de los BP o las operaciones de los WS, de acuerdo con las relaciones entre los conceptos que las describen. Los principales estándares de ontologías son RDF (Resource Description Framework) (W3C, 2004b), DAML (DARPA Agent Markup Language)(Hendler y McGuiness, 2000), OIL (Ontology Interface Layer)(Fensel y cols., 2000) y OWL (Web Ontology Language)(W3C, 2004a).

2.1.4. Lenguajes de descripción de ontologías

El crecimiento descontrolado en la generación y distribución del conocimiento en internet ha propiciado la especificación formal de la información a través de modelos computacionalmente comprensibles como las ontologías. En consecuencia, los lenguajes primitivos para el intercambio de información tales como KIF, Ontolingua, Frame Logic, entre otros han sido reemplazados por lenguajes de descripción más avanzados como RDF, DAML, OIL y OWL, los cuales permiten explotar las características de la web de una forma más eficiente y facilitan los procesos de razonamiento, inferencia y evaluación automática sobre las descripciones de los recursos.

La gestión de WS y BP no ha sido ajena a esta tendencia con respecto al manejo de la información, de hecho, en los últimos años las organizaciones han promovido la utilización de ontologías para el enriquecimiento semántico de los documentos descriptores de sus recursos (WS y BP) como una medida para reducir la ambigüedad y facilitar el entendimiento común de los objetivos del negocio en todos los niveles empresariales.

Por esta razón, en la actualidad existen numerosos lenguajes estándares de descripción de

ontologías empleados tanto para el manejo de la información disponible en la web como para la gestión semántica de BP. A continuación se presenta una breve descripción de los lenguajes más conocidos y utilizados para la descripción de ontologías.

- **Entorno para la descripción de recursos (RDF):** RDF es un lenguaje de propósito general desarrollado por el W3C (World Wide Web Consortium) con el objeto de representar información de los recursos de la web a través de un marco común que facilita el intercambio de información entre aplicaciones. Para esto, emplea simples enunciados compuestos por propiedades y valores (tripletas RDF) acerca de los recursos. A pesar de su marco común de intercambio de información, este lenguaje no permite precisar cuándo las descripciones de los recursos se refieren a un tipo o clase específica de un dominio(Manola y Miller, 2004). Por esta razón, requiere de un mecanismo para definir vocabularios especializados en los enunciados, el cual se conoce como "Lenguaje de Descripción de Vocabularios en RDF" (RDFS por sus siglas en inglés)(Brickley y Guha, 2002).

- **Esquema RDF (RDFS):** RDFS es una extensión semántica de RDF, que define un mecanismo para describir las propiedades y las clases de los recursos RDF a través de terminologías que facilitan el modelado de objetos con una semántica definida para especificar propiedades aplicables a diferentes clases de objetos(Brickley y Guha, 2002). Adicionalmente, posibilita la instanciación de objetos a partir de clases con propiedades y restricciones, y un conjunto de términos que permiten: construir expresiones o sentencias acerca de los recursos; representar sus propiedades; y establecer, mediante relaciones taxonómicas, las jerarquías de generalización entre las clases.

  Por otra parte, este lenguaje admite la construcción de ontologías simples sobre las cuales pueden realizarse consultas y razonamientos automáticos; sin embargo no es lo suficientemente expresivo como para crear ontologías complejas, debido a que solo permite inferencias sobre la herencia de propiedades, dejando de lado la capacidad para generar axiomas (Corcho y Gómez-Pérez, 2000; Silva-Muñoz, 2002; Ossenbruggen, Hardman, y Rutledge, 2002).

  Por esta razón, la comunidad científica ha desarrollado otros lenguajes como OIL(Ontology Inference Layer), DAML-ONT (DARPA Agent Markup Language - Lenguaje de Marcas de Agentes de DARPA) y DAML+OIL, los cuales ofrecen una mayor expresividad. OIL presenta una mejor interoperabilidad semántica sobre los recursos web que RDF, puesto que emplea un modelo para la representación del conocimiento basado en lógica descriptiva (axiomas y reglas) y taxonomías de clases y atributos; sin embargo posee baja expresividad para declarar axiomas y soportar dominios concretos. Por su parte, DAML-OIL fue desarrollado como una extensión de XML y RDF para extender el nivel de expresividad de RDFS a través de una unión de los anteriores lenguajes y las características de OIL; lo cual le permite alejarse un poco del modelo basado en marcos y potenciar la lógica descriptiva (Bechhofer, Goble, y Horrocks, 2001).

- **Lenguaje para ontologías web (OWL):** OWL es un lenguaje de marcado semántico creado por el W3C para publicar y compartir ontologías en la web. Respecto a sus predecesores (RDF, OIL y DAML-ONT), OWL añade vocabulario para describir propiedades y clases con el fin de establecer relaciones adicionales entre las clases (por ejemplo de disyunción) y atribuir ciertas propiedades a las relaciones (por ejemplo de cardinalidad, simetría, transitividad o relaciones inversas). Adicionalmente, OWL incluye toda la capacidad expresiva de RDFS y la extiende con la posibilidad de utilizar expresiones lógicas (Castells, 2003).

  Si bien, estas características hacen de OWL un lenguaje poderoso en expresión, también generan dificultades en cuanto a la complejidad y capacidad de decisión computacional (García Sánchez, 2008; McGuinness y Harmelen, 2004), lo cual limita la capacidad de los razonadores para inferir nuevo conocimiento a partir de sentencias dadas. Por esta razón, fueron creadas tres variantes o sub-lenguajes de OWL que proveen expresividad incremental con propiedades computacionales diferentes: OWL Lite, OWL DL y OWL Full (García Sánchez, 2008). En la actualidad, OWL es el lenguaje de representación de ontologías más utilizado en el contexto de la web semántica debido a que se constituyó en un estándar avalado por el W3C.

- **Lenguaje para el modelado de servicios web (WSML)** WSML es un lenguaje de modelado que establece un mecanismo formal para describir los elementos definidos en la ontología formal WSMO (Web Service Modelling Ontology), la cual fue desarrollada por un conjunto de empresas coordinadas por la institución de investigación de empresas digitales (DERI por sus siglas en inglés)(Roman y cols., 2005). La ontología WSMO proporciona un modelo conceptual para la descripción de diversos aspectos relacionados con servicios web semánticos e identifica cuatro elementos de alto nivel que agrupan los conceptos principales que deben describirse en la definición de un servicio web semántico: *las ontologías* que proporcionan la terminología usada por otros elementos de WSMO; *los servicios web* que representan entidades computacionales capaces de proporcionar acceso a los servicios; *las metas* que describen aspectos relacionados con los deseos de los usuarios respecto a la funcionalidad solicitada; y *los mediadores* cuyo objetivo es el manejo automático de problemas de interoperabilidad entre diferentes elementos de WSMO (Roman y cols., 2005).

  A pesar de que WSMO proporciona el modelo conceptual para describir los anteriores elementos (entre los que se encuentran las ontologías), carece de una formalización real de los conceptos para que puedan ser computacionalmente procesables. Es por esto que el W3C recomienda utilizar WSML como el lenguaje formal para la escritura, almacenamiento y comunicación de los conceptos definidos en WSMO (Bruijin y Heymans, 2006).

  WSML está basado en diferentes lógicas formales denominadas: Lógica de Descripción, Lógica de Primer Orden y Lógica de Programación; las cuales son utilizadas para el modelado de servicios web semánticos (SWS). A partir de estas lógicas formales

fueron definidas cinco variantes de WSML: WSML-Core, WSML-DL, WSML-Flight, WSML-Rule y WSML-Full; cada una con distinto nivel de expresividad lógica. Entre las cinco variantes, la más apropiada para describir ontologías es WSML-Flight ya que es la variante más expresiva y combina tanto la lógica de descripción como la de primer orden con una complejidad computacional manejable (Bruijin y Heymans, 2006).

Una de las ventajas más importantes que presenta WSML es que está especificada a través de una sintaxis legible, basada en XML y RDF, para el intercambio de datos entre servicios, lo cual le brinda interoperabilidad con otras aplicaciones compatibles con RDF. Además, WSML discrimina claramente entre sintaxis conceptual y sintaxis de expresión lógica, permitiendo modelar distintivamente los elementos conceptuales de WSMO (servicio web, ontologías, metas y mediadores), en una manera independiente de la descripción de restricciones adicionales y axiomas(Lausen, Bruijn, Polleres, y Fensel, 2005).

2.1.5. Procesos de negocio (BP)

Los procesos de negocio (BP por sus siglas en inglés) están relacionados con el concepto de orquestación, debido a que permiten organizar componentes software dentro de una estructura de ejecución centralizada. Dichas estructuras de ejecución pueden entenderse como servicios complejos con funciones suministradas por diferentes servicios componentes, los cuales existen en la web y están dinámicamente integrados para garantizar una tarea de negocio más compleja (Mongiello y Castelluccia, 2006).

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 (The-WfMC, 1999). En este sentido los BP pueden modelarse a través de la captura de secuencias ordenadas de actividades e información de apoyo que una empresa realiza para conseguir sus objetivos de negocio (White, 2006). En la actualidad existe una gran variedad de lenguajes y formalismos para modelar BP como puede observarse en la sección 2.1.7.

2.1.6. Patrones de flujo de control de BP

El flujo de control 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. El flujo de control está ligado a los nodos de control como por ejemplo XOR (Split-Join), AND (Split-Join) y OR (Split-Join), los cuales definen relaciones de dependencia (paralelismo, sincronización, selección, entre otras) entre las tareas de un BP.

En este sentido los patrones de flujo de control corresponden a estructuras que representan comportamientos específicos de los BP (la sección 3.1.3 describe los patrones relevantes para

la presente propuesta científica). Estos patrones fueron inicialmente introducidos por la comunidad científica, liderada por el Dr. Van der Aalst (van der Aalst, Hofstede, Kiepuszewski, y Barros, 2003), la cual inicialmente propuso 20 patrones de flujo de control que posteriormente fueron incrementados a 43 (Russell, H.M, Hofstede, van der Aalst, y Mulyar, 2006). En la figura 2.1 se muestra por ejemplo un patrón denominado *"Parallel Split"* representado a través de redes de petri coloreadas.

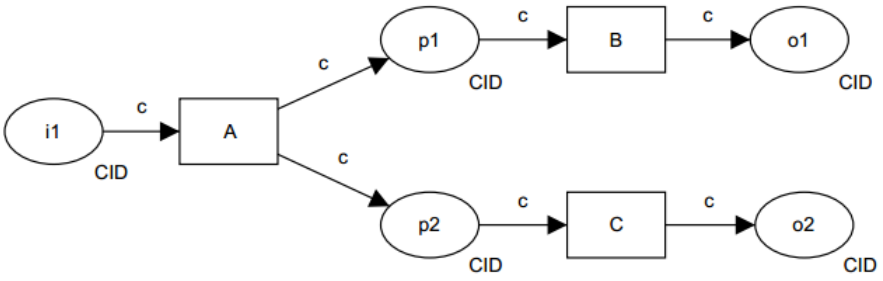


Figura 2.1: Patrón Parallel Split representado con redes de petri. Fuente (Russell y cols., 2006).

2.1.7. Lenguajes de modelado de procesos de negocio

Esta sección presenta una clasificación de los lenguajes de modelado de BP de acuerdo al trabajo desarrollado por Ko, et.al (Ko, Lee, y Lee, 2009), el cual organiza los estándares de modelado de acuerdo al ciclo de vida BPM (Business Process Modeling) . Tal y como se puede observar en la figura 2.2 este ciclo de vida está compuesto por las fases de: diseño del BP, configuración del sistema, despliegue del proceso, y diagnóstico.

Lenguajes gráficos

Los lenguajes gráficos permiten modelar BP de manera sencilla e intuitiva incluso para usuarios no relacionados con lenguajes de programación, debido a que ofrece herramientas que representan BP como diagramas gráficos a partir de transiciones, tareas, y flujos de control. En esta categoría pueden encontrarse los siguientes lenguajes:

- **Diagramas de activiad de UML (UML-D):** los diagramas de actividad corresponden a una herramienta de modelado del estándar UML (Unified Modeling Language) propuesto por el grupo OMG (Object Management Group), el cual 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

Figura 2.2: Ciclo de vida BPM. Fuente (van der Aalst, Hofstede, y Weske, 2003).

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 (Keller, Nuttgens, y Scheer, 1992). Este lenguaje está centrado en la descripción de procesos a nivel de la lógica de negocio (no necesariamente en un nivel formal de especificación) con el fin de facilitar su uso y comprensión por parte de los profesionales de gestión de negocios.

  Este lenguaje hereda su nombre de los diagramas que representan la estructura del flujo de control de los procesos como una cadena de eventos y funciones, es decir, una cadena de procesos dirigida por eventos (van der Aalst, 1999). Algunas de las ventajas de EPC son: abarca aspectos y descripciones detalladas de las unidades de organización con sus respectivas funciones de negocio y recursos materiales; y posee un lenguaje rico en el modelado de dominio, organización, perspectivas funcionales y 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 para el 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.

Una característica importante de BPMN es su capacidad para consolidarse 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). Adicionalmente, esta notación facilita 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), puedan expresarse visualmente mediante una notación estándar (White, 2006).

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 flujo de control. 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 pueden desplegarse en motores de ejecución con el fin de ejecutar cada una de sus tareas de acuerdo a unas reglas de flujo de control 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 de acuerdo con la semántica predefinida. Los procesos BPML están basados 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 están: la capacidad para permitir que los programadores puedan centrarse en la definición del proceso y sus secuencias de ejecución (flujo de control) sin preocuparse por lenguajes de bajo nivel; y la facilidad para implementar reusabilidad y escalabilidad en los sistemas de gestión de BP (BPMS). En cuanto a sus debilidades puede nombrarse que BPML no permite evidenciar fácilmente los componentes temporales de un proceso; la baja aceptación en el mercado y que actualmente es obsoleto a pesar de haber sido constituido como un estándar (Dubray, 2008).

- **Lenguaje de ejecución de procesos de negocio (BPEL):** BPEL es un lenguaje de modelado de BP ejecutables basado en XML y estandarizado por OASIS (Jordan

y Alves, 2007), el cual 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 sin revelar su comportamiento interno (Corrales, Grigori, y Bouzeghoub, 2006); y un proceso ejecutable, que 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.

Estructuralmente, un proceso BPEL está dividido 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 (Estero y Medina Bulo, 2008).

- **Lenguaje de workflow YAWL:** YAWL (Yet Another Workflow Language, por sus siglas en inglés) fue desarrollado a partir de la definición de patrones de flujo de control introducidos por Rusell y cols., (Russell y cols., 2006). Dichos patrones están representados a través de redes de petri y especifican diferentes comportamientos de ejecución que un BP puede realizar en su ciclo de vida. 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 a través de XML Schema, XPath, y XQuery. No obstante, a pesar de sus excelentes características solo es utilizado masivamente en el entorno académico y no en la industria.

Lenguajes de intercambio

Los lenguajes de intercambio proveen portabilidad de información entre BP creados en diferentes lenguajes de modelado.

- **Metamodelo para la definición de procesos (BPDM):** BPDM es un estándar definido por el grupo OMG, que ofrece un entorno para representar modelos de BP independientemente de la notación o metodología utilizada. Para esto establece 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 ofrece una sintaxis XML para facilitar la integración, almacenamiento y transformación de modelos de BP entre diferentes notaciones. Por esta razón BPDM actúa como un estándar traductor multilenguaje que provee una abstracción de los elementos básicos y comunes entre lenguajes como BPEL, BPMN, XPDL, XLANG, WSFL y UML-AD (Ko y cols., 2009).

- **Lenguaje XML para la definición de procesos (XPDL):** XPDL es un lenguaje estándar propuesto por la WfMC en su modelo de referencia (Hollingsworth, 1995); está basado en XML y soporta todos los aspectos enunciados en la especificación BPMN, incluyendo las descripciones gráficas y las propiedades ejecutables (WfMC, 2011). Entre sus principales características se pueden destacar: la alta estandarización y aceptación tanto por la comunidad académica como por la industria; y las grandes facilidades de conversión entre diferentes lenguajes.

Lenguajes semánticos

Son aquellos lenguajes de modelado de BP que admiten la adición de anotaciones semánticas para enriquecer características como por ejemplo, los nombres y las interfaces (entradas y salidas) de las tareas de los BP.

- **Entorno semántico para servicios web (OWL-S):** OWL-S es un protocolo de modelado de WS semánticos basado en ontologías OWL (Gater, Grigori, y Bouzeghoub, 2010), el cual facilita que usuarios finales y expertos de software puedan descubrir, invocar, componer y vigilar los recursos web con un alto grado de automatización. Para esto OWL-S ofrece 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 (Martin y cols., 2004).

- **Ontología para modelar procesos de negocio (BPMO):** BPMO es un lenguaje semántico que propone un modelo semántico para enriquecer la descripción de los BP en un alto nivel de abstracción y facilitar la integración de los lenguajes gráficos con las especificaciones técnicas de implementación de los BP(Z. Yan, Cimpian, Mazzara, y Zaremba, 2007). Adicionalmente, BPMO implementa una ontología para representar artefactos en varias metodologías de modelado de los BP a través de una representación unificada. Dicha ontología fue propuesta por el proyecto de la unión europea denominado SUPER (Semantics Utilized for Process Management within and between Enterprises)[1] 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. Entre las características más importantes de este lenguaje se pueden nombrar: la facilidad de traducción entre diferentes lenguajes; la 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

[1] El principal objetivo del proyecto SUPER fue llevar la gestión de BP (BPM) desde el nivel de las tecnologías de la Información (TI) hacia el nivel de negocio, para lo cual desarrolló herramientas que permiten el despliegue de la gestión de BP 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/

académico, su documentación es escasa, y su soporte se descontinuó con la finalización del proyecto SUPER a finales del año 2009.

2.1.8. Formalismos para el modelado de procesos de negocio

Esta sección presenta los principales formalismos (lenguajes formales) para el modelado de BP, los cuales a diferencia de los lenguajes presentados en la sección anterior, que permiten modelar aspectos técnicos de los BP, proveen una fundamentación teórica y académica para definir BP sin ambigüedad y con facilidad de análisis matemáticos.

- **Autómatas de estados finitos (FSA):** los 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 (Black, 2008). Generalmente los FSA son representados 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 (Daciuk, Watson, Mihov, y Watson, 2000). En los diagramas de transición de estados las transiciones están etiquetadas con mensajes obtenidos de un conjunto de mensajes, y los estados finales marcados con círculos concéntricos. Formalmente un FSA se puede definir como $FSA = (\sum, S, s_0, \delta, F)$, donde $\sum$ 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).

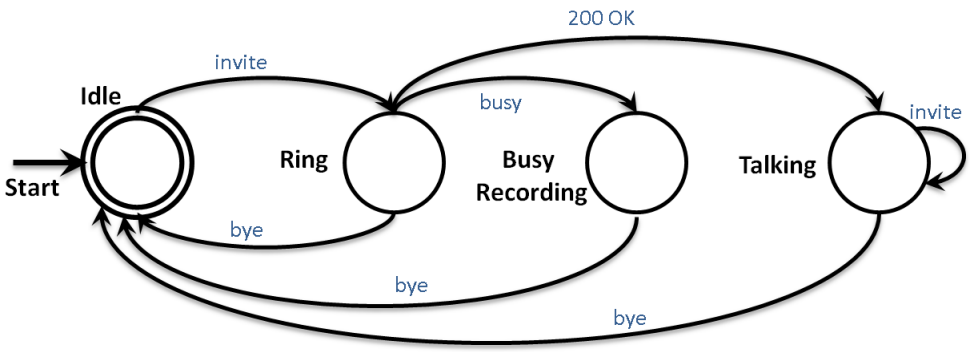


Figura 2.3: Ejemplo de un FSA para el servicio básico de llamada. Fuente Propia.

Por último cabe anotar que los FSA en su forma original no pueden representar ejecuciones paralelaS como lo permiten otros lenguajes más expresivos, por ejemplo las redes de petri (Corrales, 2008). La figura 2.3 muestra un ejemplo del proceso *"servicio básico de llamada"* representado como un autómata finito con cinco estados denotados por circulos (*idle, ring, busy recording* y *talking*) y un conjunto de transiciones que representan los cambios de estado (*invite, 200 OK, busy* y *bye*). En este ejemplo, el estado

idle es un estado aceptador o final; por esta razón aparece representado por dos círculos concéntricos.

- **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 y bipartito, que ofrece una semántica formal para análisis matemáticos; y por lo tanto es adecuada para modelar, simular y estudiar BP (Z. Yan y cols., 2007). No obstante, el modelo clásico de las redes de petri tiene problemas para representar muchos sistemas que se encuentran en logística, producción, comunicaciones, manufactura flexible y procesamiento de información, para lo cual requiere algunas extensiones como por ejemplo las redes de workflows y los patrones de workflows (Russell y cols., 2006).

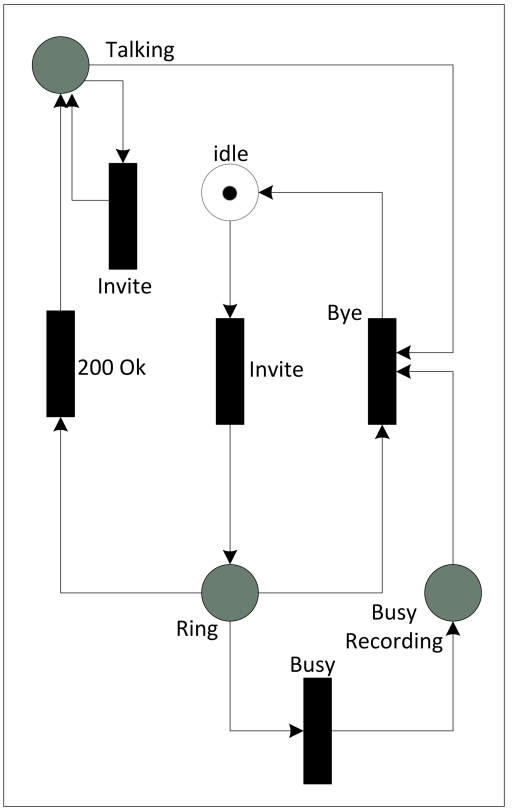


Figura 2.4: Ejemplo de una red de petri. Fuente (Benavides, Enriquez, Ramirez, Figueroa, y Corrales, 2012).

Formalmente una red de petri se puede definir como $PN = (P, T, A, W, M0)$, donde $P = (p_1, p_2, ..., p_m)$ es un conjunto finito de lugares, $T = (t_1, t_2, ..., t_n)$ un conjunto finito de transiciones, $A \subseteq (P \times T) \cup (T \times P)$ el conjunto de aristas (relación de flujo), $W : A \rightarrow (1, 2, 3, ...)$ una función de peso y $M0 : P \rightarrow (0, 1, 2, 3, ...)$ 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 2.4 puede observarse un ejemplo de la red de petri para el servicio básico de llamada. En este ejemplo, los círculos representan los estados *idle, ring, busy, recording* y *talking*; y los rectángulos a las transiciones *invite, 200 OK, busy* y *bye*.

- **Algebra de proceso (π-calculus)** las álgebras de procesos también conocidas como π-calculus, fueron introducidas por el matemático Robin Milner en los 90 y corresponden a una familia de métodos para modelar formalmente sistemas concurrentes. El lenguaje π-calculus ofrece un entorno de representación, simulación, análisis y verificación de sistemas de comunicación móvil y 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 (Havey, 2005)), las cuales pueden enlazarse en secuencia, en paralelo, en caminos condicionales, o de manera recursiva. En este sentido, algunos estudios han demostrado que el lenguaje π-calculus está bien adaptado para modelar BP clásicos en cuanto a coreografía y orquestación (Z. Yan y cols., 2007).

- **Grafos de proceso:** los grafos constituyen una herramienta matemática de modelado formal que utiliza poderosas estructuras de datos para representar objetos y conceptos de manera genérica; están constituidos por nodos que expresan operaciones o cálculos, y aristas o enlaces entre ellos que representan dependencia de datos (Hendrickson y Leland, 1995). 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 $V \times V$ (conjunto de aristas).

  Adicionalmente, los grafos permiten modelar estructuras con alto nivel de abstracción, y la representación de relaciones de causalidad entre componentes, flujos de datos, flujo de control, y concurrencia en la ejecución de procedimientos paralelos. Por esta razón son considerados 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. Este tipo de grafos son conocidos como *"grafos de proceso"*, en los cuales existen nodos simples que tienen una arista de entrada y otra de salida y nodos de flujo de control 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 evitan conflictos mientras mantienen la concurrencia.

En la figura 2.5 puede observarse el grafo del servicio de llamada básica, el cual contiene diez nodos: un nodo de inicio, un nodo de fin, cuatro nodos de tareas básicas (*invite, ring, busy recording, invite* y *talking*) y tres nodos conectores que permiten controlar el flujo de ejecución (*XORS*, *XORJ* y *XORJ*).

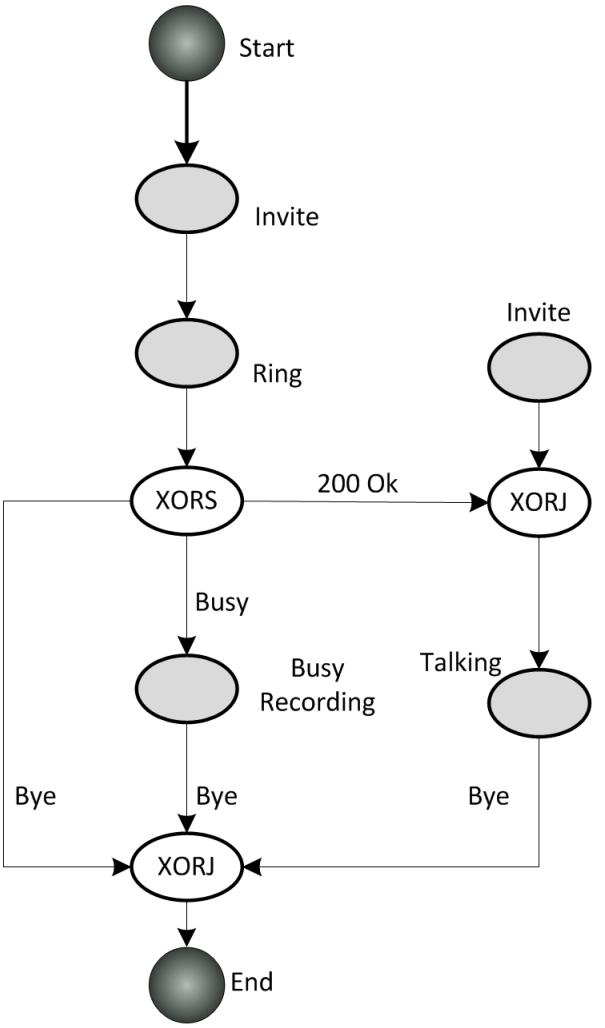


Figura 2.5: Ejemplo de un grafo de proceso. Fuente (Benavides y cols., 2012).

Entre las propiedades de los grafos como mecanismo formal de modelado de BP, se destacan su simplicidad, alto grado de abstracción y la existencia de múltiples herramientas matemáticas para su análisis. Sin embargo, justamente debido al alto grado de abstracción y simplicidad pueden presentar algunos problemas de expresividad como por ejemplo, la representación de flujos de datos en el cual las redes de petri resultan

más apropiadas.

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. Esta sección realiza un estudio acerca del estado actual de conocimiento alrededor de la temática de descubrimiento de BP y clasifica las técnicas para este propósito en cuatro niveles: interfaces, semántica, estructura, y comportamiento.

2.2.1. Descubrimiento de BP basado en interfaces

El descubrimiento de BP en este nivel tiene en cuenta parámetros relacionados con las interfaces (entradas y salidas) de las tareas que que constituyen un BP, y de los WS involucrados. En este caso, los BP pueden estar almacenados en registros como UDDI y ebXML, en los cuales el descubrimiento de BP puede llevarse a cabo a través de las interfaces de descripción WSDL. Por lo tanto, el descubrimiento tiene encuenta palabras clave, y tablas de correspondencia de parejas. Los principales trabajos en este nivel son presentados a continuación:

Stroulia y Wang., (Stroulia y Wang, 2005) discuten un conjunto de métodos para estimar la similitud de especificaciones de interfaces WSDL para lo cual utilizan la base de datos léxica *"WordNet"* (Miller, 1995). WordNet permite encontrar relaciones léxicas entre dos cadenas de texto que representan a los identificadores, las 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 y cols., (Kokash y cols., 2006) 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 en el caso anterior solo están centrados en relaciones léxicas y estructurales de los conceptos encontrados en las interfaces de los BP; y dejan de lado las relaciones de comportamiento y estructura de los BP.

Xu, Liu y Wu., (Xu, Liu, y Wu, 2007) presentan un mecanismo en tres capas para el descubrimiento de BP: primero, la capa de componentes relacionada con información macroscópica de las unidades de negocio y sus interrelaciones; segundo, la capa de operaciones 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 las coincidencias entre las entradas y salidas de las operaciones de negocio. Finalmente, para evaluar los resultados modelan su

solución a través de una máquina de estados la cual es comparada con los requisitos, de un analista de negocios, expresados con el lenguaje PSL (Property Specification Language). El enfoque de este artículo está a un nivel más alto que el técnico, es decir, centrado básicamente en unidades de negocio definidas por analistas de negocio; y por lo tanto deja de lado los aspectos técnicos de los BP como el comportamiento y la estructura.

Koschmider, Hornung y Oberweis., (Koschmider, Hornung, y Oberweis, 2011) presentan un editor de BP asistido por búsqueda de procesos completos o partes de los mismos, con el fin de facilitar su reúso en la construcción de nuevos BP. Para esto utilizan dos métodos de búsqueda: el primero, denominado búsqueda básica, permite indexar BP de acuerdo a las palabras clave relacionadas con sus atributos representados en un modelo de espacio vectorial (VSM por sus siglas en inglés) y un modelo booleano; y el segundo, 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 ejecutadas en las cadenas de texto que representan a los BP. A pesar de sus aportes en cuanto al uso de un modelo de espacio vectorial, su sistema de búsqueda no tiene en cuenta la estructura y el comportamiento en cuanto al flujo de control; ni tampoco las técnicas semánticas sobre las operaciones.

Gonçalves da Silva, Ferreira y cols., (Gonçalves da Silva y cols., 2011) proponen un método denominado DynamiCoS para descubrir correspondencias exactas y parciales entre WS. Para esto, extraen información de las entradas, salidas, precondiciones, efectos y metas de los WS; y las convierten en anotaciones semánticas que posteriormente son almacenadas en un registro denominado jUDDI (jUDDI es una versión mejorada del registro UDDI). En esta propuesta solo descubren servicios simples, es decir, no 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 pueden nombrarse, los motores de búsqueda de WS en internet, los cuales a partir de palabras clave introducidas por un usuario en una página Web, exploran al interior de las 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], Service Repository[5], MyExperiment[6], BioCataloge[7] y Programable Web[8].

2.2.2. Descubrimiento de BP basado en semántica

El nivel de descubrimiento de BP basado en semántica está centrado en la capacidad de los sistemas de recuperación de BP y WS para inferir acerca de los conceptos sobre

[2]Disponible en la dirección: http://www.webservicex.net
[3]Disponible en la dirección: http://webservices.seekda.com
[4]Disponible en la dirección: http://www.webservicelist.com
[5]Disponible en la dirección: http://www.service-repository.com
[6]Disponible en la dirección: http://www.myexperiment.org
[7]Disponible en la dirección: http://www.biocatalogue.org
[8]Disponible en la direcci: http://www.programmableweb.com

los cuales el cliente está realizando su búsqueda. Para esto, el nivel semántico considera criterios ontológicos que tienen en cuenta el significado y las relaciones entre los conceptos; y define 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 son descritos a continuación:

Châtel (Châtel, 2007) describe un modelo para la localización semántica de WS y BP, el cual considera 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, es soportado por un marco semántico basado en lenguajes y tecnologías comunes de la semántica como por ejemplo: SAWSDL(Semantic Annotations for WSDL) (Kopecky, Vitvar, Bournez, y Farrell, 2007), 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 ejecuta funciones de mapeo entre SAWSDL y UDDI. Por otra parte, puede nombrarse que este trabajo no considera las declaraciones del servicio como un todo sino que está centrado únicamente en las operaciones.

Klusch y cols., (Klusch y cols., 2006) presentan un método de correspondencia híbrida para WS semánticos que complementa el razonamiento lógico con correspondencia aproximada basada en cálculos de similitud sintáctica. Este método es implementado a través de un prototipo denominado OWLS-MX el cual toma como consulta un servicio descrito en el lenguaje OWL-S y retorna un conjunto ordenado de servicios relevantes clasificados teniendo en cuenta 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 con el contenido del perfil de interfaces (entradas y salidas) de servicios OWL-S.

Lin y Arpinar (Lin y Arpinar, 2006) presentan una técnica de descubrimiento de WS, basada en pre y post condiciones, en la cual las capacidades de los servicios son expresadas 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 (Okkyung y Sangyong, 2008) presentan un motor de búsqueda de servicios basado en la combinación de reglas con ontologías. Para esto proponen un método que personaliza las búsquedas de WS ya que ejecuta 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 computacional debida a la gran variedad de algoritmos de activación utilizados, lo cual incrementa el consumo de tiempo en las búsquedas.

Kiefer, Bernstein y cols., (Kiefer, Bernstein, Lee, Klein, y Stocker, 2007) 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 que permite 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. Los principales problemas de este trabajo son: actúa en dependencia del dominio, y no realiza correspondencia ni alineación de ontologías.

Verma, Akkiraju y cols., (Verma, Akkiraju, Goodwin, Doshi, y Lee, 2004) 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 su simplicidad debido a 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 los trabajos del 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 de BP basado en estructura

Este nivel compara la estructura de los BP, generalmente representados a través de formalismos de modelado (sección 2.1.8). Los formalismos son útiles en este nivel ya que facilitan el análisis estructural de los BP utilizando técnicas matemáticas, como por ejemplo el isomorfismo de grafos que permite evidenciar si dos estructuras son iguales o similares. Los principales trabajos en este nivel son:

Corrales, Grigori y cols., (Corrales y cols., 2010) comparan BP utilizando un algoritmo de correspondencia estructural que busca grafos de proceso dentro de un repositorio de acuerdo a un grafo de consulta. Dicho algoritmo es conocido 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. Adicionalmente, el trabajo de Corrales, Grigori y cols., utiliza como soporte para el algoritmo de comparación estructural, un algoritmo de comparación léxica entre cada una de las actividades o tareas de los BP. El algoritmo léxico confronta las etiquetas de los nombres de las tareas y 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 (flujo de control), 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 compra los nombres de las actividades.

Eshuis y Grefen., (Eshuis y Grefen, 2007) establecen mecanismos para comparar BP especificados en el lenguaje BPEL, utilizando como criterios 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 BP son similares pero no idénticos; e inclusión, cuando uno de los BP está contenido en otro.

Wombacher y Li., (Wombacher y Li, 2010) presentan dos medidas importantes para evaluar la similitud de BP. La primera medida (basada en mecanismos similares a los presentados en el nivel de interfaces) está fundamentada en una distancia lingüística conocida como distancia Hamming, la cual permite calcular el menor número de operaciones de edición de símbolos (sustituciones, inserciones y eliminaciones). Los símbolos se encuentran en las cadenas de texto que identifican a las actividades y se comparan con el fin de hacerlas similares a las cadenas de texto de las actividades de un BP 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 y cols., (Corrales y cols., 2010), calcula una distancia de edición para que un grafo de proceso (localizado en un repositorio o registro) se asemeje tanto como sea posible a un grafo de proceso de consulta.

Sakr y Awad., (Sakr y Awad, 2010) 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 implementa consultas considerando los diferentes tipos de nodos y la complejidad de las aristas. Adicionalmente, 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 están aplicados en el descubrimiento.

Los trabajos del nivel estructural están centrados en la comparación de BP a través de algoritmos estructurales basados generalmente en isomorfismo de grafos, pero no tienen en cuenta el comportamiento (flujo de control 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 de BP basado en comportamiento

La clasificación de los trabajos en este nivel está definida por varios aspectos que pueden determinar el comportamiento de un BP, como por ejemplo el intercambio de mensajes dentro de las actividades, los registros de ejecución histórica y el flujo de control. A continuación

son descritos los principales trabajos en el nivel de comportamiento.

Los trabajos de Wombacher, Mahleko y cols., (Wombacher, Fankhauser, Mahleko, y Neuhold, 2004; Mahleko y Wombacher, 2006) presentan un mecanismo de correspondencia en el cual los BP son representados a través de 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 es representada por la comparación de dos aFSA, las cuales son similares si: existe al menos un camino (secuencias de mensajes) común entre los estados de inicio y fin, y cumple con un conjunto de mensajes obligatorios y opcionales. Los caminos son representados a través de secuencias de ejecución representadas como listas de n-gramas, en las cuales cada n-grama está relacionado con un estado en particular y una lista de ellos puede representar una abstracción del conjunto de estados localizados en una secuencia de ejecución.

Shen y Su., (Shen y Su, 2005) presentan un modelo que asocia mensajes intercambiados entre participantes con actividades ejecutadas dentro de un servicio. Las actividades son catalogadas en un perfil descrito en OWL-S, los servicios modelados a través de autómatas finitos no deterministas, y la recuperación de los mismos ejecutada a través de un lenguaje de consultas que permite expresar propiedades temporales y semánticas de los servicios.

Yun, Yan y cols., (Yun, Yan, Liu, y Yu, 2008) 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 y cols., está basado en el nivel de comportamiento en cuanto al orden de ejecución de las actividades, pero no tiene en cuenta el flujo de control, ni tampoco la semántica de las interfaces de las actividades.

Goedertier y cols., (Goedertier y cols., 2011) hacen un estudio de los principales algoritmos para descubrir BP utilizando registros de eventos que representan 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 y cols., los principales algoritmos en esta área son:

- Algoritmo alfa (α) (van der Aalst y cols., 2004) y sus variaciones $(\alpha+, \alpha++)$ (de Medeiros, Dongen, y van der Aalst, 2004): este algoritmo propone un orden binario local de las relaciones detectadas dentro de un registro de eventos. Sin embargo, es incapaz de descubrir ciclos cortos no locales y actividades duplicadas, ni tampoco tiene en cuenta la frecuencia de las secuencias binaras que ocurren en los registros de eventos.
- Algoritmo Heuristics Miner (de Medeiros y cols., 2004): este método representa vi-

sualmente 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 (de Medeiros, Weijters, y van der Aalst, 2007): 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 (Artificially Generating Negative Events) (Goedertier, Martens, Vanthienen, y Baesens, 2009): 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 eventos analizando las restricciones frecuentes que se encuentran en ellos. Posteriormente el algoritmo utiliza dichas restricciones para detectar dependencia local, dependencia no local y relaciones de paralelismo existentes entre pares de actividades. Infortunadamente, los registros de eventos rara vez contienen información acerca de transiciones no permitidas, para lo cual el algoritmo AGNEs debe generar dichos registros artificialmente, de manera muy similar a como se hace en la actividad de aprendizaje para el contexto de minería de datos.

Como puede observarse, los trabajos del nivel de comportamiento no tienen en cuenta la semántica atribuida a las tareas internas del BP y no aplican técnicas semánticas para detectar las diferentes estructuras del flujo de control.

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 (Sellami y cols., 2008; Nayak y Lee, 2007; Sapkota, 2005) 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 basado en la semántica de comportamiento, es decir la detección de patrones de flujo de control y sus relaciones semánticas; y en segundo lugar un mecanismo de refinamiento de correspondencia aproximada que implementa una comparación estructural de BP, 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 realizó un estudio conceptual y un estado actual del conocimiento sobre el descubrimiento de BP. Inicialmente, describió las principales tecnologías como BP, WS, web semántica, y los lenguajes y formalismos de modelado de BP. Luego, presentó el estado actual de conocimiento acerca de las técnicas de descubrimiento de BP y WS, las cuales fueron clasificadas 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 flujo de control, registros de eventos y secuencias de mensajes.

Capítulo 3

Pre-correspondencia de procesos de negocio

La investigación científica descrita en este libro propone un entorno de recuperación de BP 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.

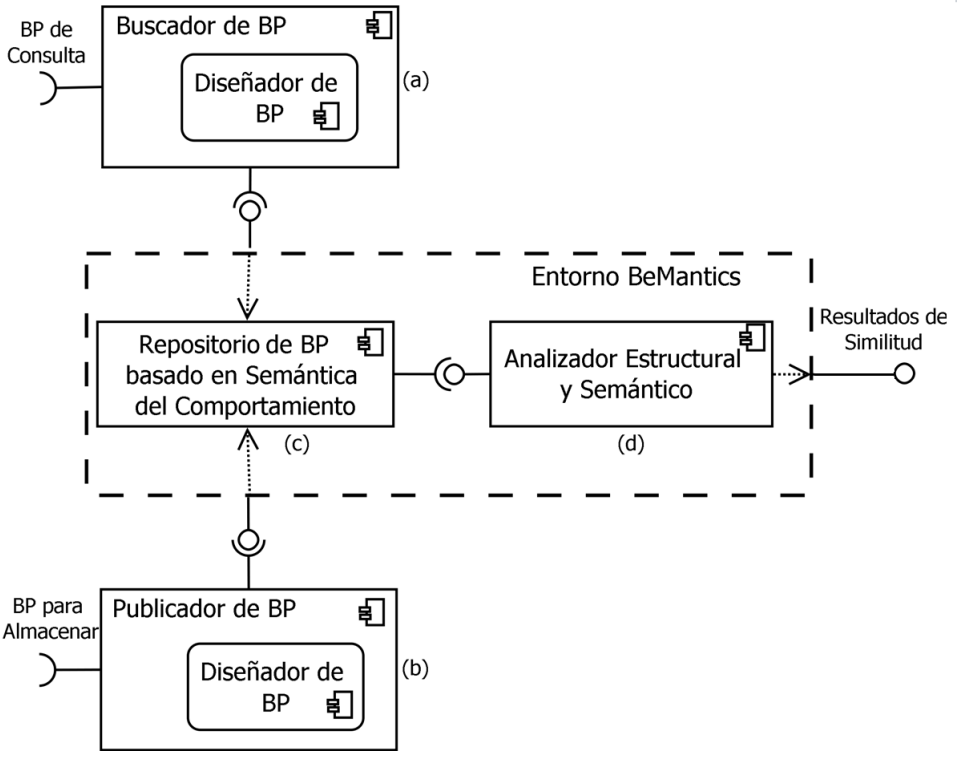


Figura 3.1: Arquitectura de referencia para el entorno BeMantics. Fuente propia.

En la figura 3.1 se presenta la arquitectura de referencia de BeMantics en la cual pueden observarse sus dos módulos principales, el módulo de pre-correspondencia denotado como "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; y el módulo de correspondencia denominado "analizador estructural y semántico" (figura 3.1 (d)), para 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 de 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 del repositorio y de los BP de consulta respectivamente.

El presente capítulo presenta inicialmente conceptos preliminares necesarios para el correcto entendimiento del entorno BeMantics y a continuación describe el funcionamiento del repositorio y de los módulos de diseño de BP. El cuarto módulo, el analizador estructural y semántico (figura 3.1 (d)) será discutido con detalle en el capítulo 4.

3.1. Conceptos preliminares

Esta sección desarrolla los conceptos preliminares alrededor de los cuales gira la investigación presentada en este libro. Inicialmente, describe algunas técnicas de correspondencia de grafos y posteriormente presenta los patrones de flujo de control utilizados para definir el comportamiento de los BP.

3.1.1. Teoría de grafos

En la sección 2.1.8 (página 17) fueron presentados los principales formalismos de modelado de BP, entre los cuales se destacaron los grafos de proceso, debido a su simplicidad y capacidad expresiva para entornos de BP. La presente sección hace un resumen de la teoría de grafos y su aplicación en el entorno BeMantics utilizando técnicas matemáticas como el isomorfismo y la indexación de grafos.

Definiciones básicas

Los grafos constituyen una herramienta matemática para modelar sistemas a través de estructuras de datos expresivas y genéricas, que permiten la representación de objetos y conceptos. Éstos están constituidos por nodos, que representan operaciones o cálculos; y aristas o enlaces entre ellos que expresan dependencia de datos(Hendrickson y Leland, 1995). Por esta razón, los grafos pueden considerarse como excelentes herramientas para modelar BP, en los cuales un nodo puede representar una tarea y las aristas el intercambio de mensajes y datos entre los nodos. De este modo los grafos facilitan el modelado de estructuras con alto nivel de abstracción, y la representación de relaciones de causalidad entre componentes,

flujos de datos, flujo de control y concurrencia en la ejecución de procedimientos paralelos.

En este libro, son denominados como *"grafos de proceso"* aquellos grafos que representan BP. En este tipo de grafos existen nodos simples, que tienen una arista de entrada y otra de salida; y nodos de control, que representan puntos de flujo de control que pueden tener múltiples entradas y salidas dependiendo de su tipo (OR, XOR, AND etc). De esta manera los flujos de ejecución de un grafo de proceso evitan conflictos a la vez que mantienen la concurrencia.

Un grafo G es un par (V, E) donde V es un conjunto (llamado conjunto de nodos) y E un subconjunto de VxV (conjunto de aristas). Gráficamente los nodos son representados por puntos y las aristas por líneas que los unen. Un nodo puede tener cero o más aristas, pero toda arista debe unir exactamente dos nodos. Se denomina orden de un grafo a su número de nodos y es denotado como $|V|$. Si $|V|$ es finito se dice que el grafo es finito.

- ***Aristas:*** si la arista carece de dirección se denota indistintamente $\{a, b\}$ o $\{b, a\}$, siendo a y b los nodos que une. Si $\{a, b\}$ es una arista, entonces los nodos a y b son sus extremos.
- ***Nodos:*** dos nodos v,w son adyacentes si $\{v, w\} \in A$ (o sea, si existe una arista entre ellos). Se denomina grado de un nodo al número de aristas de las que es extremo. Un nodo es "par" o "impar" según sea su grado.
- ***Caminos:*** sean $x, y \in V$, se dice que existe un camino en G de x a y si existe una sucesión finita no vacía de aristas $\{x, v_1\}, \{v_1, v_2\}, \ldots, \{v_n, y\}$. En este caso:
  - x e y son conocidos como los extremos del camino
  - El número de aristas del camino es denominada longitud del camino.
  - Si los nodos no se repiten el camino es propio o simple.
  - Si hay un camino no simple entre dos nodos, también habrá un camino simple entre ellos.
  - Cuando los dos extremos de un camino son iguales, el camino es denominado circuito o camino cerrado.
  - Un ciclo es un circuito simple
  - Un nodo es accesible desde un nodo a a un nodo b si existe un camino entre ellos. Todo nodo es accesible respecto a si mismo

Clasificación de grafos

Los grafos se pueden clasificar en dos grupos: dirigidos y no dirigidos. En un *grafo no dirigido* el par de nodos que representan a una arista no está ordenado. Por lo tanto, los pares (v_1, v_2) y (v_2, v_1) corresponden a la misma arista. Formalmente, un grafo no dirigido $G = (V, E)$ está definido por:

- $V \neq 0$

- $E \subseteq \{x \in p(v) : |x| = 2\}$

En contraste, en un *grafo dirigido* cada arista está representada por un par ordenado de nodos; por lo tanto los pares (v_1, v_2) y (v_2, v_1) corresponden a dos aristas diferentes. Formalmente, un grafo dirigido $G = (V, E)$ está definido por:

- $V \neq 0$
- $E \subseteq \{(a, b) \in VXV : a \neq b\}$

En este caso, dada una arista (a, b), se tiene que a representa su nodo inicial y b su nodo final. A continuación, son descritos tres ejemplos de grafos dirigidos y no dirigidos.

- *Ejemplo N° 1:*
  $G_1 = (V_1, A_1)$
  $V_1 = \{1, 2, 3, 4\}$
  $A_1 = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$
- *Ejemplo N° 2:*
  $G_2 = (V_2, A_2)$
  $V_2 = \{1, 2, 3, 4, 5, 6\}$
  $A_2 = \{(1, 2), (1, 3), (2, 4), (2, 5), (3, 6)\}$
- *Ejemplo N° 3:*
  $G_3 = (V_3, A_3)$
  $V_3 = \{1, 2, 3\}$
  $A_3 = \{< 1, 2 >, < 2, 1 >, < 2, 3 >\}$

Gráficamente las tres estructuras de nodos y aristas de los ejemplos pueden representarse como lo muestra la figura 3.2:

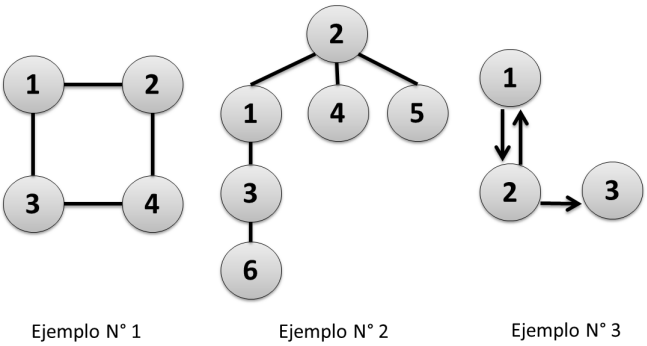


Figura 3.2: Ejemplos de grafos dirigidos y no dirigidos. Fuente propia.

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 (Rosen, 2006). Formalmente un isomorfismo entre dos grafos G y G'; es un mapeo biyectivo $f : V \rightarrow V'$ 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 $\beta(e) = \beta'(e')$.

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 también tener en cuenta si uno de ellos está contenido en otro grafo (es decir un sub-grafo). Un ejemplo de esto puede observarse en la validación que realiza el entorno BeMantics para determinar si un grafo contiene una sub-estructura específica de comportamiento. Formalmente; $f : V \rightarrow V'$ es un isomorfismo entre dos grafos G y G'; y G' un sub-grafo 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, en la figura 3.3 puede observarse un grafo G y un sub-grafo G_1 de G.

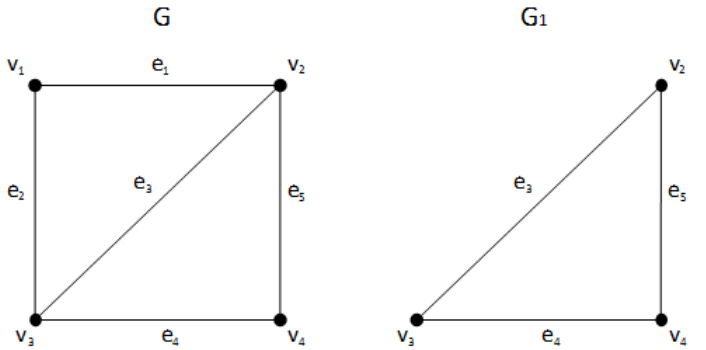


Figura 3.3: Ejemplo de sub-grafo. Fuente propia.

Indexación de grafos

La indexación de BP define índices para identificar los modelos almacenados en un repositorio de manera tal que facilite realizar búsquedas rápidas de los procesos o de colecciones de los mismos. En la actualidad existen diferentes clases de índices centrados en características de los BP a clasificar, 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, son presentados algunos trabajos orientados a la perspectiva de la búsqueda, recuperación y minería de grafos[1].

Los trabajos de Yan y Han (X. Yan y Han, 2002, 2003) están enfocados en la recuperación de grafos basada en minería de patrones de grafos frecuentes, en la cual un grafo es considerado frecuente si su frecuencia de ocurrencia en un conjunto de datos dado no es menor que un umbral mínimo. Para esto utilizan algoritmos de búsqueda en profundidad (DFS - Depth First Search) y descomposición de subestructuras de patrones. Otro trabajo enfocado en los patrones de grafos frecuentes es presentado por Zhu y cols., (Zhu, Yan, Han, y Yu, 2007) en el cual describen las técnicas de minería de patrones de grafos utilizando un algoritmo de poda denominado *"gprune"* para reducir el espacio de búsqueda de acuerdo a restricciones estructurales con el fin de reforzar el proceso de minería.

Los trabajos de Giugno, Shasha y cols., (Giugno y Shasha, 2002; Ferro y cols., 2007); Srinivasa y Singh (Srinivasa y Singh, 2005) y Yan, Yu y Han., (X. Yan, Yu, y Han, 2004) definen métodos de indexación en los cuales a partir de una base de datos de grafos encuentran todos los grafos que contienen o están contenidos en un grafo de consulta. De esta manera es posible ejecutar mecanismos de 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 y cols., (Jin, Wang, Wu, La Rosa, y Ter Hofstede, 2010) 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 que lo contienen. Este conjunto de procesos es posteriormente filtrado utilizando índices que permiten generar listas de candidatos. Luego aplica una técnica de isomorfismo de grafos sobre este conjunto de candidatos, a través de una adaptación del algoritmo de Ullmann (Ullmann, 1976). Para esto, el trabajo de Jin y cols., integra características de varios repositorios de grafos con algunas de las técnicas de búsqueda y minería de grafos descritos anteriormente.

El entorno BeMantics descrito en este libro adapta el algoritmo GraphBlast (Ferro y cols., 2007) para implementar un mecanismo de indexación de estructuras frecuentes de grafos denominadas patrones de flujo de control. Estas estructuras están estrechamente relacionadas con las relaciones de dependencia (paralelismo, sincronización, selección, entre otras) existentes entre las tareas de un BP y los flujos de ejecución en los cuales están involucradas. Los patrones de flujo de control fueron introducidos por la comunidad investigativa liderada por el Dr. Van der Aalst (van der Aalst, Hofstede, Kiepuszewski, y Barros, 2003) en la cual fue propuesta una lista inicial de 20 patrones básicos que posteriormente se incrementaron a 43 en el estudio de Rusell y cols., (Russell y cols., 2006). Adicionalmente, GraphBlast ofrece un alto rendimiento debido a que implementa el algoritmo de isomorfismo de grafos llamado VF2 (P. Cordella, Foggia, Sansone, y Vento, 2004), el cual presenta mayor rendimiento en el tiempo de respuesta en comparación con el algoritmo de Ullmann.

[1]En general la minería de grafos puede describirse 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

3.1.2. Repositorios de procesos de negocio

Esta sección presenta los principales trabajos de investigación que estudian el almacenamiento de BP. El trabajo de Yan, Dijkman y Grefen., (Z. Yan, Dijkman, y Grefen, 2012) presenta los resultados de un análisis detallado de 16 repositorios de BP comparados de acuerdo a sus características de datos, funcionalidad y gestión, teniendo en cuenta los aspectos de almacenamiento, recuperación, integración e indexación. Dichos trabajos son resumidos a continuación.

- **Librería de procesos de negocio (BPL):** BPL (Kaczmarek, Konstantinov, Ma, Wieloch, y Zebrowski, 2008) es una librería de BP que provee operaciones estándar de un sistema de gestión de bases de datos, como: crear, leer, actualizar, borrar artefactos, y soportar transacciones a través una herramienta gráfica. Adicionalmente ofrece dos tipos de consulta, el primero centrado en un sistema de bases de datos tradicional, y el segundo basado en consultas semánticas con inferencia sobre la información contenida en una base de datos relacional. BPL es un proyecto muy amplio que comprende una plataforma para la gestión de BP, sin embargo, muchas de sus funcionalidades están centradas en una arquitectura cerrada la cual utiliza herramientas propietarias del proyecto europeo *SUPER*.

- **Repositorio ebXML (Negocios electrónicos utilizando lenguaje de marcado extensible):** ebXML (Park, Kim, You, y Choi, 2007) es un repositorio incluido en la familia de estándares ebXML como una iniciativa de OASIS (Organization for the Advancement of Structured Information Standards) y UN/CEFACT (United Nations Centre for Trade Facilitation and Electronic Business). Este repositorio permite almacenar prácticamente cualquier tipo de datos incluyendo: descripciones de servicios web, documentos y datos XML, y datos binarios (imágenes, archivos de sonido, datos de video, archivos de aplicaciones ejecutables, entre otros). Por medio de este registro es posible adicionar meta información a los datos ingresados en el repositorio, y hacer búsquedas y clasificaciones mediante un mecanismo avanzado de consultas XML o SQL (Zaremba, Wall, y Browne, 2002). ebXML ejecuta consultas SQL y XML, pero carece de capacidades para soportar consultas semánticas.

- **Repositorio IBM para BPEL:** el repositorio de IBM para BPEL (Vanhatalo, 2004) forma parte del proyecto Business Process Integration and Automation (BPIA) del laboratorio de investigaciones de IBM en Zurich. El repositorio almacena procesos de negocio BPEL y otros documentos XML en sistemas de archivos. Permite hacer consultas a los archivos XML como objetos EMF (Eclipse Modeling Framework) utilizando un lenguaje de consultas orientado a objetos (OCL), así como también proporciona operaciones básicas de almacenamiento como crear, leer, escribir y borrar con el fin de manipular los datos almacenados en XML como objetos Java.

- **Repositorio XML para la gestión integrada de procesos (IPM):** el repositorio IPM (Choi, Kim, y Jang, 2007) gestiona BP a través del ciclo de vida, para lo cual cuenta con funciones avanzadas en la gestión de configuración y versiones. El intercambio de la información de los BP con el repositorio está condicionado por el lenguaje de

modelado IPM-EPDL (IPM- Executable Process Definition Language) y las consultas por el lenguaje de consultas IPM-PQL (IPM - Process Query Language). Este repositorio es una interesante opción para almacenar modelos específicos de los BP y ejecutar sus instancias. No obstante, tanto el lenguaje de consultas como el de modelado son privativos.

- **Repositorio repoX:** Este repositorio (Song, Miller, y Arpinar, 2001) es parte del proyecto METEOR,[2] almacena modelos de BP representados en el lenguaje XML, los cualers pueden ser intercambiados usando documentos XML en un formato predefinido y almacenados en una base de datos orientada a objetos. RepoX tiene en cuenta los aspectos del flujo de control de los BP junto con sus datos y los roles que son autorizados para ejecutar tareas de los BP. Sin embargo, no cuenta con una notación específica, por lo que la descripción de los BP tiene que realizarse a través de DTD la cual tiene bastantes limitaciones en la definición de elementos de contenido; y por otra parte, las consultas están basadas en XQuery el cual permite extraer información en documentos XML pero no permite realizar consultas semánticas.

- **Repositorio oryx:** Oryx (Decker, Overdick, y Weske, 2008) es una herramienta de modelado de procesos basada en web que soporta búsqueda de usuarios, creación, almacenamiento y actualización de modelos de procesos en línea. La herramienta utiliza un repositorio para almacenar los modelos de BP que son creados en él. ORYX se enfoca principalmente en el aspecto del flujo de control y almacena modelos de procesos específicos de compañías, además soporta muchas notaciones de modelado de procesos, incluyendo: BPMN, EPC, redes de petri, redes de workflows, entre otros. Además, no solamente se enfoca en las funciones básicas de almacenamiento (crear, leer, escribir y borrar), sino que implementa funciones de importación y exportación de modelos en formatos ERDF, JSON, PNML, XPDL y XHTML.

- **Repositorio de procesos variantes (PVR):** PVR (Lu, Sadiq, y Governatori, 2009) este repositorio provee mecanismos para tratar con diferentes variantes de los modelos de BP en tiempo de ejecución. Para este propósito, contiene funcionalidades de almacenamiento que incorporan restricciones de variaciones permitidas de los modelos de BP. La información histórica acerca de las variaciones realizadas en tiempo de ejecución también está almacenada con el fin de mejorar el modelado de BP. PVR soporta diversas características de los BP como: actividades, flujo de control, datos, recursos, aspectos de monitoreo y autorización; y almacena procesos específicos de compañías, incluyendo sus instancias e información histórica acerca de ejecución. A pesar, de que este repositorio contiene un lenguaje de consultas de definiciones e instancias, no posee un lenguaje específico para el almacenamiento y recuperación.

- **Entorno para consultar BP en la fase de modelado :** este entorno (Markovic, Pereira, y Stojanovic, 2008) contiene mecanismos de consultas expresivas para la fase de modelado de BP, que soportan características de los BP como: actividades, flujo de control,

[2]El proyecto (METEOR) fue desarrollado para la gestión de BP a gran escala, en entornos de computación heterogéneos y multi-empresariales.

datos, objetivos, información de recursos y autorización. Además, brinda al usuario experto una librería de artefactos de BP que contiene patrones, fragmentos, y referencias de BP soportados en un alto nivel de abstracción y almacenados en un repositorio de ontologías RDF.

- **Repositorio prosero:** prosero (Elhadad, Balaban, y Sturm, 2008) consta de un repositorio de modelos de BP de referencia (RMR), un repositorio de de servicios web (WSR) y un repositorio de modelos del usuario (CMR). A través de estos repositorios permite almacenar modelos de BP de referencia y modelos de BP específicos de compañías e instancias de los mismos. Prosero soporta características de los BP como: actividades, flujo de control, datos, recursos, autorización y estructura organizacional; además almacena BP descritos en BPMN, y ejecuta procesos BPEL debido a que incluye un generador de BPEL para transformar modelos BPMN a BPEL.

- **Apromore:** Apromore (La Rosa y cols., 2011) es un repositorio avanzado que mantiene, analiza y reutiliza grandes colecciones de modelos de BP. Además, es una plataforma de código abierto, implementada de acuerdo a la arquitectura SOA y por lo tanto accesible a través de servicios web. La representación de los procesos está basada en un formato canónico que utiliza EPC y BPMN como lenguajes de modelado. Actualmente existe una versión de un prototipo disponible en la red que contiene funcionalidades básicas como: importar, exportar, buscar, clasificar y comparar modelos de BP.

3.1.3. Patrones de flujo de control

En la sección 2.1.6 fue explicada brevemente la relación entre los patrones de flujo de control y el comportamiento en cuanto a los flujos de ejecución que pueden tomar lugar al interior de un BP teniendo en cuenta las relaciones de dependencia entre sus tareas. La presente sección hace una selección de doce patrones de flujo de control utilizados como base para determinar el comportamiento de procesos BPMO.

El trabajo de Rusell y cols., (Russell y cols., 2006) hace un estudio de los patrones de flujo de control y su compatibilidad en diferentes lenguajes de modelado de BP. No obstante, entre el conjunto de lenguajes estudiados no está considerado el lenguaje BPMO que fue el lenguaje seleccionado para la presente propuesta. Por lo tanto, en esta sección es presentado un resumen del estudio de viabilidad de la implementación de los patrones en la notación BPMO.

Por ejemplo, uno de los patrones es conocido como unión parcial estructurada (*Structured Partial Join*) cuya funcionalidad consiste en generar un flujo de salida únicamente cuando cumpla un subgrupo determinado de actividades. Este patrón puede observarse en la figura 3.4(a) en la cual el elemento “2-out of 3-join” solamente genera un flujo de salida cuando cumple un subgrupo de dos actividades de un total de tres. Este patrón no existe directamente en BPMO, pero analizando su funcionamiento puede encontrarse un equivalente por medio de una relación de composición a partir de los elementos de BPMO (figura 3.4(b)).

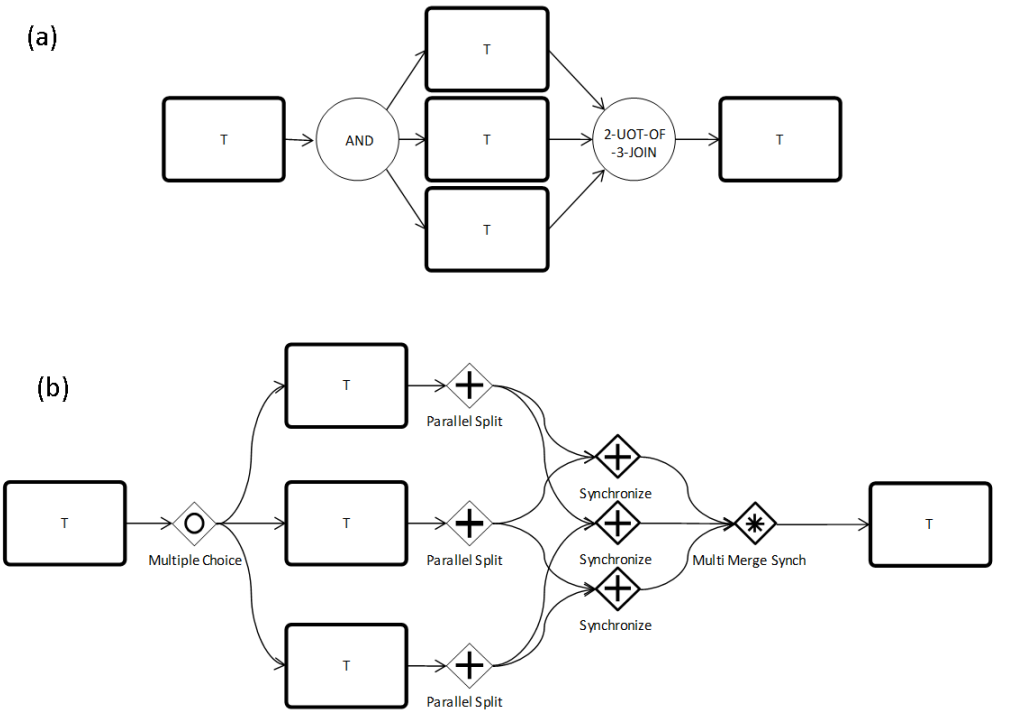


Figura 3.4: (a) Patrón de unión parcial estructurada, (b) Implementación del patrón en BPMO. Fuente propia.

Patrones de flujo de control en BPMO

A continuación se realiza una descripción de cada uno de los doce patrones de flujo de control viables para su implementación en BPMO. Dicho conjunto de patrones de flujo de control es un subconjunto de los cuarenta y tres definidos por el trabajo de de Rusell y cols., (Russell y cols., 2006), el cual describe detalladamente la funcionalidad de cada uno de ellos.

- **Patrón secuencia (sequence):** este patrón sirve como el bloque fundamental para los modelos BP. Consiste en una secuencia de mínimo tres actividades de tipo tarea "T", las cuales, de acuerdo a la especificación BPMO 1.4 pueden ser de tipo "tarea manual", "tarea de servicio web" o "tarea de meta". En este patrón, el flujo de control conecta cada tarea precedente con la siguiente, como se puede apreciar en las flechas continuas de la figura 3.5.

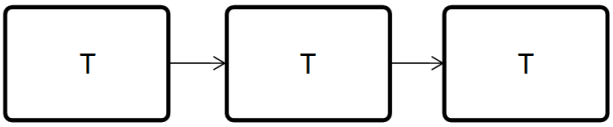


Figura 3.5: Patrón secuencia con tres tareas. Fuente propia.

- **Patrón división paralela (parallel split):** este patrón permite ejecutar paralelismo en los flujos de ejecución al dividir el flujo de control en dos o más flujos salientes que ejecutan actividades simultáneamente. La figura 3.6 presenta un nodo de tipo tarea que una vez ejecutado genera un flujo de control entrante hacia una compuerta "parallel split", la cual divide el flujo de control en dos ramas que posteriormente ejecutan en forma simultánea las dos tareas siguientes.

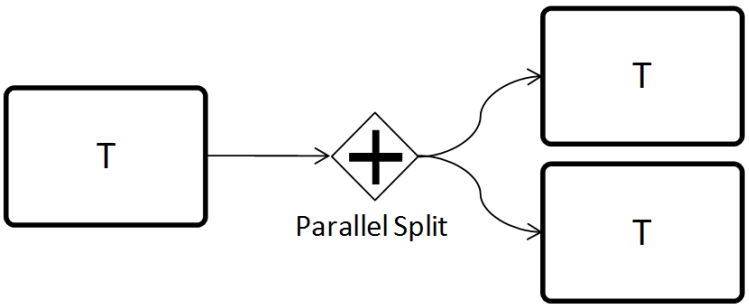


Figura 3.6: Patrón división paralela. Fuente propia.

- **Patrón sincronización (synchronize):** este patrón permite la sincronización de flujos de ejecución mediante la convergencia del flujo de control de dos o más ramas en una sola, de tal manera que el flujo de control de la rama saliente ocurra solo cuando las actividades de cada rama sean ejecutadas. La figura 3.7 presenta dos actividades de tipo tarea que terminan su ejecución en momentos diferentes, generando flujos de control asíncronos. Por lo tanto la compuerta "synchronize" espera a que finalice la ejecución de todas las tareas entrantes para generar el flujo de control de salida.

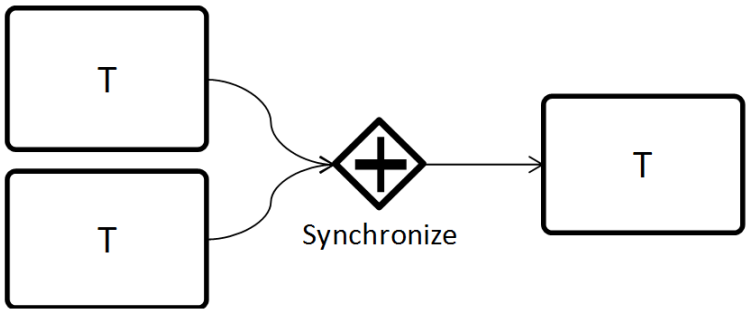


Figura 3.7: Patrón sincronización. Fuente propia.

- **Patrón selección exclusiva (exclusive choice):** este patrón permite la divergencia del flujo de control de una rama en dos o más ramas de tal manera que el flujo de control sea direccionado a una actividad específica dependiendo del resultado de una actividad precedente o algunos datos específicos que representan la existencia de una decisión

para el enrutamiento del flujo de control. La figura 3.8 muestra una actividad de tipo tarea que después de ejecutarse genera un flujo de control de salida hacia una compuerta "Exclusive Choice" para luego direccionar la ejecución por solo una de las ramas salientes dependiendo de los resultados de la tarea inicial. El patrón de selección exclusiva es uno de los más utilizados para casos condicionales, en los cuales la selección de un flujo u otro en los BP depende de unas condiciones evaluadas en la compuerta "Exclusive Choice".

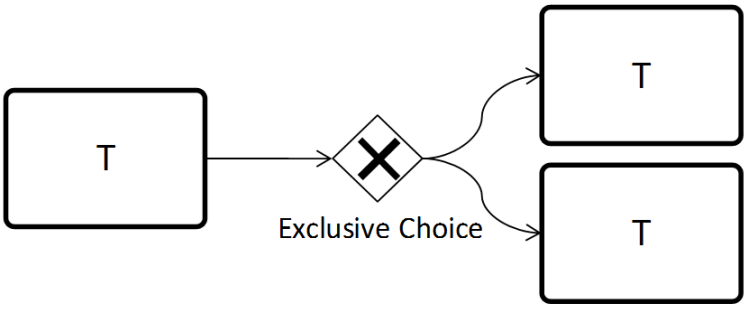


Figura 3.8: Patrón selección exclusiva. Fuente propia.

- **Patrón combinación simple (simple merge):** este patrón permite la convergencia asíncrona del flujo de control de dos o más ramas en una sola, de tal manera que el flujo de ejecución se pasa a una sola rama de salida. La figura 3.9 presenta dos actividades de tipo tarea en las cuales, independiente del momento en que terminen su ejecución, los flujos generados hacia la compuerta "simple merge" convergen de manera asíncrona ocasionando que la tarea siguiente se ejecute de manera diferente dependiendo de cada uno de los resultados de las tareas iniciales.

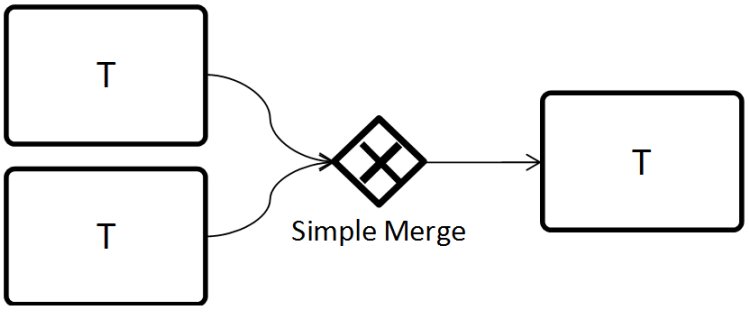


Figura 3.9: Patrón combinación simple. Fuente propia.

- **Patrón selección múltiple (multiple choice):** este patrón permite la divergencia del flujo de control de una rama en dos o más ramas, de tal manera que el flujo de ejecución sea direccionado a cada una de las ramas salientes en forma síncrona o asíncrona

dependiendo de los resultados de la tarea inicial. La figura 3.10 presenta una actividad de tipo tarea, que después de ejecutarse genera un flujo de control entrante para la compuerta "multiple choice", la cual divide de manera síncrona o asíncrona el flujo de control hacia las dos tareas posteriores.

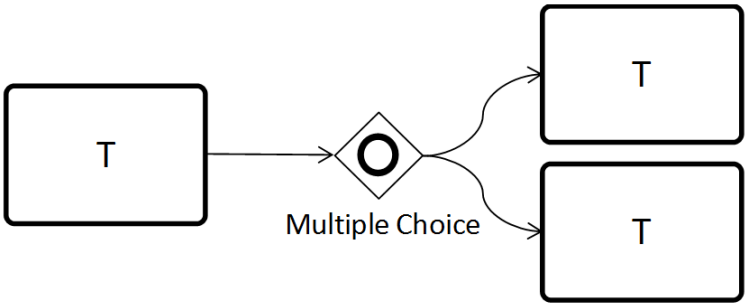


Figura 3.10: Patrón selección múltiple. Fuente propia.

- **Patrón combinación múltiple (multi merge):** este patrón permite la convergencia síncrona o asíncrona del flujo de control de dos o más ramas en una sola de tal manera que el flujo de ejecución pasa a una sola rama de salida. La figura 3.11 muestra dos actividades de tipo tarea, que independiente del momento en que terminen su ejecución cada uno de los flujos generados hacia la compuerta "multi merge" convergen de manera síncrona o asíncrona ocasionando una ejecución diferente de la tarea posterior para cada uno de los resultados de las tareas iniciales.

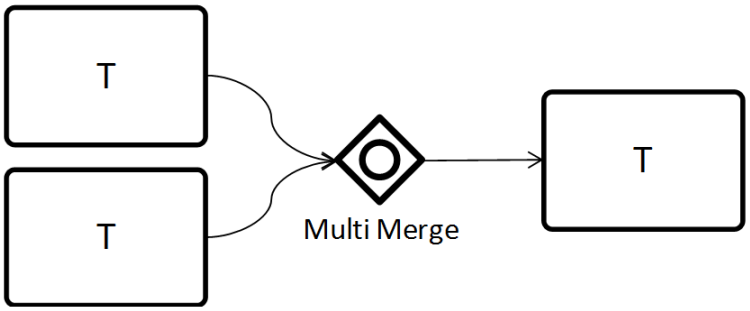


Figura 3.11: Patrón combinación simple. Fuente propia.

- **Patrón selección diferida (deferred choice):** este patrón permite la divergencia del flujo de control de una rama en dos o más ramas tal que el flujo de control sea direccionado hacia una actividad específica dependiendo del resultado de una actividad precedente. Los valores de la actividad o algunos datos específicos permiten la toma de decisión para el enrutamiento del flujo de control. La figura 3.12 presenta una actividad de tipo tarea que después de ejecutarse genera un flujo de control de salida hacia

una compuerta “deferred choice” para luego dirigir el flujo de control por una sola de las ramas salientes dependiendo de los resultados de la tarea inicial. A diferencia del patrón de selección exclusiva, en este patrón no existe una selección explícita pero si una carrera entre diferentes ramas.

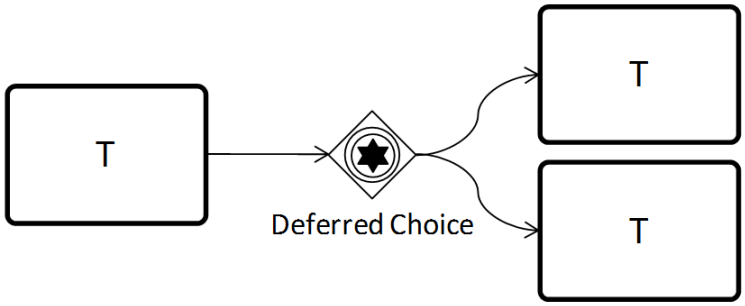


Figura 3.12: Patrón selección diferida. Fuente propia.

- **Patrón discriminante (discriminator):** este patrón permite la convergencia asíncrona del flujo de control de dos o más ramas en una sola, de forma tal, que el flujo de control se pasa a una sola rama de salida. La figura 3.13 presenta dos actividades de tipo tarea, que independiente del momento en que terminen su ejecución, cada uno de los flujos generados hacia la compuerta “discriminator” convergen de manera asíncrona. En este caso la compuerta deja pasar el flujo de ejecución generado por la primera tarea ejecutada y solo deja pasar el siguiente una vez que la primera termina su ejecución, ocasionando de esta manera una ejecución diferente para la tarea posterior.

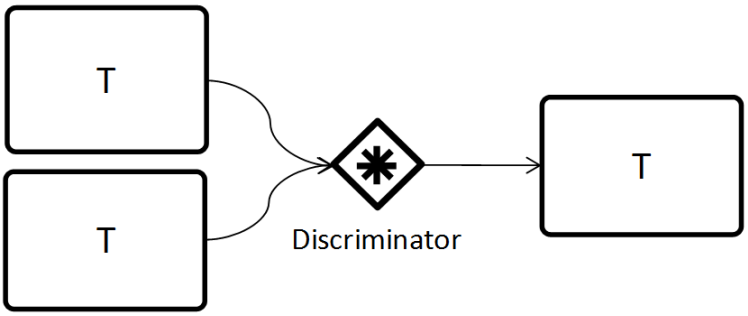


Figura 3.13: Patrón discriminante. Fuente propia.

- **Patrón de enrutamiento paralelo intercalado (interleaved parallel routing):** este patrón permite la divergencia del flujo de control en dos o más ramas dependiendo del resultado de una actividad precedente. La figura 3.14 presenta una actividad de tipo tarea que una vez ejecutada genera un flujo de control de salida hacia una compuerta “interleaved parallel routing” para luego dirigirlo por solo una de las ramas salientes

dependiendo de los resultados de la tarea inicial.

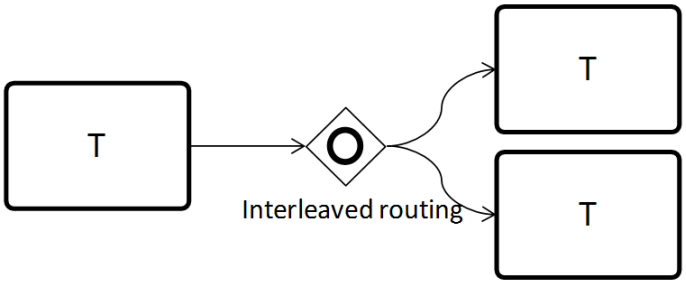


Figura 3.14: Patrón de enrutamiento paralelo intercalado. Fuente propia.

- **Patrón de combinación múltiple sincronizada (multi merge synchronize):** este patrón permite la convergencia del flujo de control de 2 o más ramas en una sola, de tal manera que el flujo de control de la rama saliente ocurra solo cuando el flujo de cada rama entrante converja una vez ejecutadas las actividades de cada rama. La figura 3.15 presenta dos actividades de tipo tarea que terminan su ejecución en momentos diferentes generando flujos de control asíncronos. Por lo tanto la compuerta "multi merge synchronization" espera a que finalice la ejecución de todas las tareas entrantes para generar el flujo de control de salida.

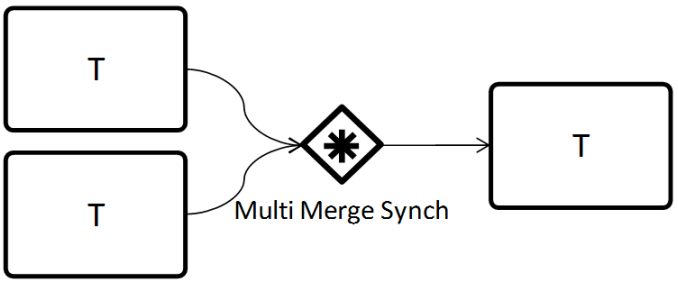


Figura 3.15: Patrón de combinación múltiple sincronizada. Fuente propia.

- **Patrón de instanciación múltiple (multiple instantiation):** este patrón permite la divergencia del flujo de control de una rama en dos o más, tal que el flujo de control se direccione a una tarea específica dependiendo del resultado de una tarea precedente. La figura 3.16 presenta una actividad de tipo tarea que después de ejecutarse genera un flujo de control de salida hacia una compuerta "multiple instantiation" para luego dirigir el flujo de control por solo una de las ramas salientes dependiendo de los resultados de la tarea inicial.

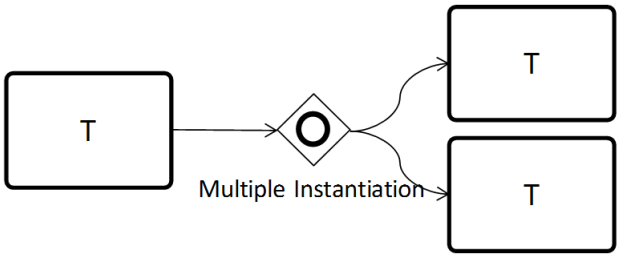


Figura 3.16: Patrón de instanciación múltiple. Fuente propia.

Clasificación de los patrones de flujo de control

La clasificación de los patrones de flujo de control, presentada en esta subsección, está basada en el trabajo de Russell y cols., (Russell y cols., 2006), el cual propone la siguiente clasificación.

- *Patrones básicos:* estos patrones están basados en las definiciones básicas de los BP, propuestas por la WfMC (WfMC, 2011); en este sentido capturan los aspectos elementales del flujo de control de los BP.
- *Patrones de ramificación y sincronización avanzada:* estos patrones se caracterizas por utilizar conceptos más complejos de ramificación y sincronización que surgen en los BP.
- *Patrones de múltiples instancias:* este grupo de patrones está enfocado en las diferentes maneras en las cuales los eventos pueden ocurrir por medio de las situaciones en donde existen múltiples hilos de ejecución activos relacionados a una misma actividad. Las múltiples instancias pueden estar declaradas en tres situaciones:
  - una actividad puede iniciar múltiples instancias de sí misma cuando es activada (se denota esta clase de actividad como una actividad de múltiples instancias).
  - una actividad dada es activada en múltiples ocasiones como consecuencia de recibir activaciones independientes (por ejemplo una parte de un ciclo o una instancia de un proceso en el cual hay muchos hilos de ejecución concurrentes como un resultado de un patrón que une el flujo de control).
  - dos o más actividades en un proceso comparten la misma definición de implementación. Como por ejemplo, el caso de múltiples instancias de actividades o el caso de la definición de un subproceso común para un bloque de actividades. En esta situación dos o más de estas actividades son iniciadas de tal forma que sus ejecuciones se superponen ya sea parcial o totalmente.

  Aunque estas tres situaciones potencialmente involucran múltiples instancias concurrentes de una actividad o un subproceso, la primera de ellas ha adquirido más importancia debido a que requiere la activación y sincronización de múltiples instancias de actividades concurrentes.

- *Patrones basados en estados:* este tipo de patrones describen la situación en la cual las soluciones son fácilmente ejecutadas en los lenguajes de BP que soportan estados. Los estados de las instancias de los BP pueden incluir una amplia colección de datos asociados con la ejecución.
- *Patrones de cancelación y terminación forzada:* estos patrones tienen variantes que utilizan el concepto de cancelación donde las instancias de actividades son activadas o desactivadas. Adicionalmente, la cancelación considera varias formas de excepción manejadas en los BP.
- *Patrones de iteración:* este tipo de patrones captura un comportamiento repetitivo de un BP.
- *Patrones de terminación:* son patrones de activación que abordan las circunstancias sobre las cuales un BP se considera completo.
- *Patrones de activación:* estos patrones están encargados de manejar las señales externas que pueden ser requeridas para iniciar ciertas actividades.

3.2. Repositorio de procesos de negocio basado en semántica del comportamiento

El repositorio es el módulo de pre-correspondencia que almacena, indexa y recupera BP utilizando técnicas que describen la semántica del comportamiento. En este contexto la semántica del comportamiento se refiere a un mecanismo de indexación que direcciona el comportamiento a través de la detección de patrones de flujo de control teniendo en cuenta su número de ocurrencias y las relaciones semánticas entre ellos (es decir las relaciones identificadas en una ontología de patrones). Para esto, el repositorio ofrece dos fases de ejecución (figura 3.17): la primera denominada fase de almacenamiento, guarda un BP en el repositorio; y la segunda denominada fase de recuperación, obtiene una lista ordenada de BP de acuerdo a los patrones de flujo de control detectados como respuesta a un BP de consulta.

La fase de almacenamiento de BP inicia cuando el usuario modela gráficamente un proceso a través del modelador gráfico WSMO Studio 0.73 (figura 3.17(b)) el cual genera un modelo del proceso en el lenguaje BPMO a partir de una la notación gráfica creada por el usuario. La figura 3.18 muestra un ejemplo gráfico de un proceso BPMO modelado a través del software WSMOS tudio, el cual representa el proceso de lanzamiento de un nuevo producto al mercado por parte de una empresa de telecomunicaciones. Este proceso contiene cuatro tareas: un evento de envío de mensaje, un evento de inicio, un evento de fin y un conector de flujo de control *"selección exclusiva"* que gestiona el orden de ejecución de las tareas subsecuentes. De esta manera, si la propuesta de un nuevo producto es aceptada, entonces el flujo de ejecución continuará hacia la ejecución de las tareas de especificación y lanzamiento del nuevo producto; de lo contrario continuará hacia la ejecución del evento para enviar el mensaje "propuesta rechazada".

Figura 3.17: Fases de ejecución del repositorio. Fuente propia.

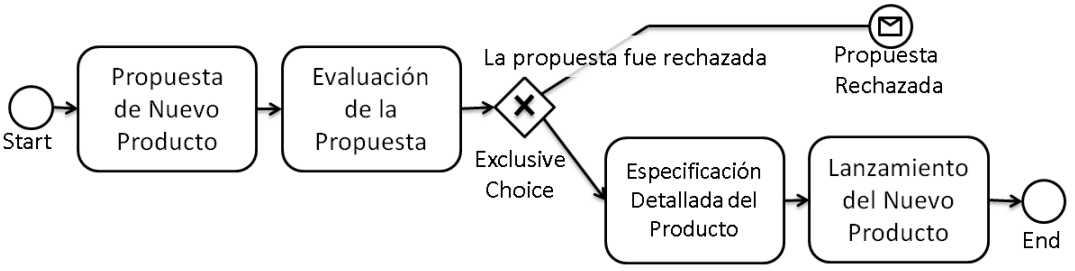


Figura 3.18: Ejemplo del proceso BPMO *"Lanzamiento de un nuevo producto"*. Fuente propia.

El siguiente paso en la fase de almacenamiento es la transformación del proceso BPMO al formalismo de los grafos con el fin de detectar ocurrencias de patrones de flujo de control, etiquetar el proceso BPMO con cada patrón encontrado y almacenarlo en el repositorio. La figura 3.19 muestra el grafo de proceso obtenido a partir del BP presentado en la figura 3.18. Este grafo contiene: tres nodos de tipo evento: *start* (1), *end* (7) y *enviar mensaje de propuesta rechazada* (8); cuatro nodos de tipo tarea correspondientes a las funciones: *propuesta de nuevo producto* (2), *evaluación de la propuesta* (3), *especificación detallada del producto* (5) y *lanzamiento del nuevo producto* (6); y un nodo de tipo conector: *XOR-Split* (4) el cual permite identificar un patrón de flujo de control.

De manera similar, en la fase de recuperación el usuario diseña gráficamente un BP de consulta el cual es transformarlo a un grafo de proceso para detectar el conjunto de patrones recurrentes. De esta manera obtiene una lista ordenada de BP almacenados en el repositorio con un conjunto similar de patrones de flujo de control.

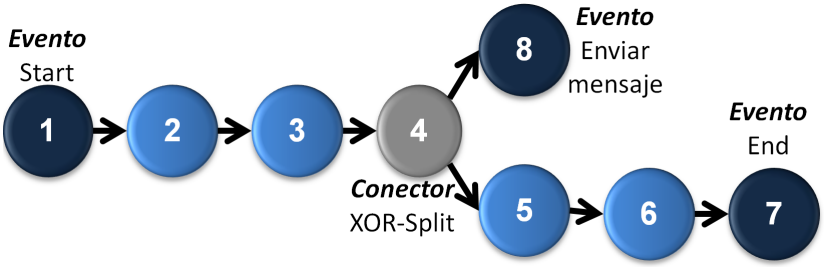


Figura 3.19: Grafo de proceso para el ejemplo *"Lanzamiento de un nuevo producto"*. Fuente propia.

Para implementar las fases descritas en los párrafos anteriores, el repositorio está compuesto por tres capas principales (figura 3.20): la primera, transformación de BP, tiene el objeto de convertir los BP a modelos formales de grafos; la segunda, análisis de patrones, detecta un conjunto recurrente de patrones en los grafos de BP; y la tercera, almacenamiento, gestiona y guarda los modelos de BP y sus correspondientes grafos.

Figura 3.20: Capas del repositorio de BP basado en semántica del comportamiento. Fuente propia.

3.2.1. Capa de transformación de BP

El repositorio almacena dos tipos de grafos útiles para posteriores análisis matemáticos de indexación y determinación de similitudes entre los BP. El primer tipo de grafos denominado grafo de proceso, 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 flujo de control. La figura 3.21 representa el procedimiento de transformación de los BP, el cual recibe como entrada un archivo WSML que contiene al proceso BPMO y entrega como salida los dos modelos de grafos (grafo de proceso y grafo TD).

El procedimiento de transformación inicia con la lectura de un archivo WSML, en el cual está contenido el proceso BPMO resultante del modelado gráfico realizado en los módulos

Figura 3.21: Transformación de un proceso BPMO a los modelos de grafos. Fuente propia.

"Publicador de BP" (figura 3.1(b)) y *"Buscador de BP"* (figura 3.1(a)). La lectura del archivo WSML es realizada a través de la librería WSMO4J (Dimitrov, Momtchev, Simov, Ognyanoff, y Konstantinov, 2006), la cual permite capturar los diferentes elementos del proceso BPMO, y a partir de ellos construir un *"grafo de proceso"* (algoritmo 1). El grafo de proceso resultante, está 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), y XOR (split, join)); y aristas sin etiquetas que los unen (Corrales y cols., 2006). Finalmente, a partir del grafo de proceso se genera un *"grafo TD"* el cual está construido evaluando los múltiples caminos (trazas) del flujo de ejecución (algoritmo 3). Los algoritmos utilizados para llevar a cabo este procedimiento son descritos a continuación.

Algoritmo 1 Algoritmo para la función convertir BPMO a grafo de proceso

Entrada: DocumentoBPMO
Salida: GrafoBPMO

```
1:  Agregar el nodo Start y el nodo End al GrafoBPMO
2:  para todo elementoBPMO hacer
3:      nodo = TransformarElementoBPMO(elementoBPMO)
4:      Agregar nodo a GrafoBPMO
5:  fin para
6:  para todo enlaceBPMO hacer
7:      arista = TransformarEnlaceBPMO(enlaceBPMO)
8:      Agregar arista a GrafoBPMO
9:  fin para
10: devolver GrafoBPMO
```

El algoritmo 1 recibe un documento WSML (Documento BPMO) y lo transforma en un grafo de proceso (Grafo BPMO) con el mapeo correspondiente entre elementos WSML y elementos del grafo (nodos y aristas). Este algoritmo, inicialmente fija los nodos de inicio y fin (Línea 1), luego obtiene todos los elementos (elemento BPMO) que componen al proceso BPMO, lo transforma a su contraparte en grafos (nodo) utilizando la función *"TranformarElementoBPMO"* y lo adiciona al grafo BPMO (líneas 2-5). Seguidamente, obtiene todos los enlaces encontrados en el documento BPMO, los transforma en aristas y los adiciona al grafo BPMO utilizando la función *"TranformarEnlaceBPMO"* (Líneas 6 - 9). Por último, retorna como resultado el grafo BPMO que corresponde al documento BPMO de entrada.

Algoritmo 2 Algoritmo para la función *"TranformarElementoBPMO"*

Entrada: elementoBPMO
Salida: nodo

1: **segun** (elementoBPMO)
2: **caso** $\in$ $(DeferredChoice \vee ExclusiveChoice \vee InterleavedParallelRouting \vee MultipleMergeSynchronise \vee Synchronization)$**:**
3: tipoNodo = Connector.XOR-Split
4: **caso** $\in$ $(Discriminator \vee SimpleMerge)$**:**
5: tipoNodo = Connector.XOR-Join
6: **caso** $\in$ MultiMerge**:**
7: tipoNodo = Connector.OR-Join
8: **caso** $\in$ MultipleChoice**:**
9: tipoNodo = Connector.OR-Split
10: **caso** $\in$ MultipleInstantiation**:**
11: tipoNodo = Connector.AND-JOIN
12: **caso** $\in$ ParallelSplit**:**
13: tipoNodo = Connector.AND-Split
14: **caso** $\in$ ErrorEvent**:**
15: tipoNodo = Event.ERROR
16: **caso** $\in$ ReceiveMessageEvent **:**
17: tipoNodo = Event.RECEIVE
18: **caso** elementoBPMO $\in$ SendMessageEvent**:**
19: tipoNodo = Event.SEND
20: **caso** elementoBPMO $\in$ TimerEvent**:**
21: tipoNodo = Event.TIMER
22: **caso** elementoBPMO $\in$ Task**:**
23: tipoNodo = Tarea
24: entradasNodo = elementoBPMO.INPUTS
25: salidasNodo = elementoBPMO.OUTPUTS
26: nombreNodo = elementoBPMO.NAME
27: nodo.TYPE = tipoNodo
28: **fin segun**
29: **devolver** nodo

El algoritmo 2 implementa la función *"TranformarElementoBPMO"* (utilizada por el algoritmo 1) la cual permite detectar el tipo de cada elemento BPMO y a partir de éste crear una clase específica de nodo (conector, función o evento) que posteriormente será agregado al grafo BPMO por el algoritmo 1. En cuanto a la función *"TranformarEnlaceBPMO"*, referenciada por este algoritmo, simplemente toma cada enlace encontrado en el documento BPMO y procede a convertirlo en una arista que une a dos nodos correspondientes a los elementos BPMO referidos por el enlace.

Las figuras 3.22 y 3.23 muestran el mapeo realizado entre elementos BPMO y elementos del grafo de proceso (nodos, conectores y aristas) llevado a cabo por el algoritmo 2.

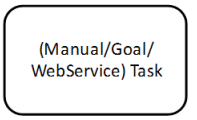
Elementos BPMO	Etiquetas de Grafos	Representaciones en los grafos
Eventos  	**Events** Los eventos se toman como nodos de tipo Evento representados por la etiqueta E	 (Start, End, Timer, Error, SendMessage, ReceiveMessage) Events
Funciones  	**Funciones** Las funciones son nodos de tipo Function representados por la etiqueta (F)	 (Manual, Goal, WebService) Tasks

Figura 3.22: Representación de las tareas de BPMO en el modelo de grafos. Fuente propia.

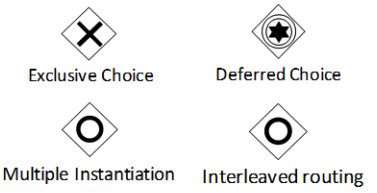
Elementos BPMO	Etiquetas de Grafos	Representaciones en los grafos
 	Las compuertas son representadas por nodos Conector (ANDS, ANDJ, ORS, ORJ, XORS XORJ)	 (DeferredChoice, ExclusiveChoice, InterleavedParallelRouting, Multiple Instantiation) Gateways
 	XOR Split se representa con la etiqueta XORS	 (Discriminator, SimpleMerge) Gateway
 	AND Split se representan con la etiqueta ANDS	 ParallelSplit Gateway
 	AND Join se representan con la etiqueta ANDJ	 (MultipleMergeSynchronise, Synchronisation) Gateways
 	OR Join se representan con la etiqueta ORJ	 Multimerge Gateway
 	OR Split se representan con la etiqueta ORS	 MultipleChoice Gateway

Figura 3.23: Representación de las compuertas de BPMO en el modelo de grafos. Fuente propia.

El último algoritmo en el procedimiento de transformación es el algoritmo de detección de trazas o TDA (algoritmo 3), el cual ejecuta la detección de patrones utilizando el software

"GraphBlast". El algoritmo TDA crea varias rutas denominadas *"trazas"*; para lo cual genera cuatro 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* almacena los nodos conectores que tienen la condición a ser evaluada; y el último denominado (*"Backtrace"*) contiene todas las trazas creadas (GrafoTD).

Algoritmo 3 Algoritmo TDA (detección de trazas)

Entrada: GrafoBPMO
Salida: GrafoTD

```
1:  i = NodoStart
2:  Adicionar i a NodosVisitados
3:  n = funcionAdyacencia(i)
4:  Adicionar n a NodosVecinos
5:  si nodosVecinos > 1 entonces
6:      Adicionar nodosVisitados a BackTrace
7:  si no, si nodosVecinos = 0 entonces
8:      FIN
9:  fin si
10: mientras nodosSinVisitar > 0 hacer
11:     j = obtenerUltimoNodoAgregado(NodosSinVisitar)
12:     si j es el ultimoNodo de GrafoBPMO entonces
13:         Adicionar j a NodosVisitados
14:         Adicionar NodosVisitados a GrafoTD
15:         si BackTrace > 0 entonces
16:             Obtener el ultimo conjunto de BackTrace
17:             NodosVisitados = GrafoTD
18:             devolver Grafo TD
19:         fin si
20:     si no
21:         Adicionar j a NodosVisitados
22:         m = funcionAdyacencia(j)
23:         Adicionar m a NodosVecinos
24:         si NodosVecinos > 1 entonces
25:             Adicionar NodosVisitados a BackTrace
26:         fin si
27:     fin si
28: fin mientras
29: devolver GrafoTD
```

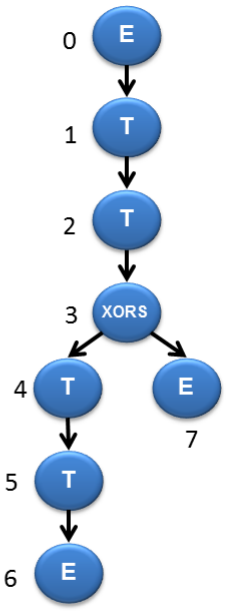
Inicialmente el algoritmo TDA toma el nodo i correspondiente al evento *"start"* del documento BPMO (es decir, el primer nodo presente en el grafo BPMO) y lo adiciona al conjunto de nodos visitados (*NodosVisitados*) (líneas 1 - 2). Luego aplica una función de adyacencia (*funcionAdjacencia()*) que retorna el nodo próximo a i que aún no haya sido visitado (nodo n). Entonces, el algoritmo toma el nodo n y lo adiciona al conjunto de nodos

vecinos (*NodosVecinos*) (líneas 3 - 4). A continuación, verifica la existencia de nodos en el conjunto *"nodosVecinos"*; y si existe al menos un nodo en este conjunto procede a adicionar los nodos visitados al conjunto *"Backtrace"* (líneas 5 - 8).

Posteriormente, el algoritmo evalúa de manera recurrente el conjunto de nodos sin visitar y dependiendo de la existencia de nodos en este conjunto ejecuta los siguientes pasos: primero obtiene el último nodo j agregado al conjunto de nodos sin visitar, si éste es el último nodo del grafo BPMO lo adiciona al conjunto de nodos visitados, el cual a su vez, es agregado al grafo TD(líneas 10 - 14); segundo, si el *"Backtrace"* contiene nodos, obtiene su último nodo y agrega el conjunto de nodos visitados al grafo TD (líneas 14 - 18); tercero, en caso que j no sea el último nodo del grafo, entonces es adicionado al conjunto de nodos visitados, el algoritmo aplica la función de adyacencia sobre j y agrega el nodo adyacente m al conjunto de nodos vecinos (líneas 15 - 23). En este caso, si el conjunto de nodos vecinos contiene más de un nodo entonces el conjunto de nodos visitados es adicionado al *"Backtrace"* (líneas 24 - 26). Finalmente, el algoritmo retorna el grafo TD generado (línea 29).

Línea	GrafoTD
1	#graph0
2	**8**
3	E
4	T
5	T
6	XORS
7	T
8	T
9	E
10	E
11	**7**
12	0 1
13	1 2
14	2 3
15	3 4
16	4 5
17	5 6
18	3 7

(a) Representación textual



(b) Representación gráfica

Figura 3.24: (a) Contenido del archivo .dat del grafo TD; y (b) Su representación gráfica. Fuente propia.

La figura 3.24 muestra un ejemplo del contenido del grafo TD generado al aplicar el algoritmo TDA (algoritmo 3) sobre el grafo de proceso del ejemplo presentado en la

figura 3.18. En este caso, el grafo TD corresponde a un documento de texto almacenado con una extensión dat (.dat), el cual está dividido en cinco secciones (figura 3.24(a)).

La primera sección (línea 1), inicia con el carácter # seguido de una cadena de texto que representa el identificador del grafo (graph#0). La segunda sección (línea 2), indica el número de nodos del grafo TD, el cual como mínimo debe ser "2" ya que si es menor no existe flujo de control alguno. La tercera sección (líneas 3 - 10), 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 flujo de control 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 flujo de control; la segunda etiqueta "T" corresponde al nodo de destino y tiene "1" como identificador numérico de flujo de control y así sucesivamente hasta llegar a la última etiqueta dentro de esta sección. La cuarta sección (línea 11), indica el número de aristas del grafo TD, el cual como mínimo debe ser "1", de acuerdo con la restricción de la segunda sección del grafo TD. La quinta y última sección (líneas 12 - 18), 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 3.24(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), y además el etiquetado correspondiente para los elementos de flujo de control del modelo BP de ejemplo *"Lanzamiento de un nuevo producto.wsml"* presentado en la figura 3.18.

3.2.2. Capa de análisis de patrones

Esta capa está encargada de detectar patrones de flujo de control en los grafos TD correspondientes al BP de consulta y a cada uno de los BP del repositorio. La figura 3.20 muestra la capa de análisis de patrones compuesta por dos sub-capas: el detector de patrones, y el organizador semántico de resultados, a través de las cuales, actúan las dos fases descritas al inicio de este capítulo.

De esta manera, en la fase de almacenamiento: el detector de patrones encuentra un conjunto de patrones en un grafo de proceso (BP del repositorio) y lo etiqueta con el identificador de cada patrón detectado con el fin de facilitar su indexación y almacenamiento; y en la fase de recuperación: el detector de patrones encuentra los patrones en un BP de consulta; y el organizador semántico de resultados encuentra y clasifica por relevancia aquellos BP del repositorio que contengan un conjunto similar de patrones con respecto al BP de consulta.

Detector de patrones

Esta sub-capa recibe como entrada un grafo de proceso, y retorna un conjunto de patrones detectados, los cuales están representados como estructuras denominadas sub-grafos. Por lo

tanto, el problema de detección de patrones es reducido a un problema de isomorfismo de sub-grafos el cual busca sub-estructuras al interior de un grafo de proceso (Ferro y cols., 2007; X. Yan y cols., 2004; Giugno y Shasha, 2002; Zhu y cols., 2007).

La investigación descrita en este libro adaptó un método de isomorfismo de sub-grafos denominado *GraphBlast* (Ferro y cols., 2007), el cual está basado en un algoritmo denominado VF2 (P. Cordella y cols., 2004), para indexar sub-estructuras contenidas en un gran conjunto de grafos (una base de datos de grafos). El algoritmo VF2 (algoritmo 4) ejecuta una técnica de isomorfismo de grafos fundamentada en un método de correspondencia determinístico, a través del cual es posible comparar diferentes grafos y determinar si un grafo está contenido en otro (sub-grafo) o si son iguales en cuanto a su estructura de nodos y aristas (isomorfo). El algoritmo es válido mientras no existan restricciones impuestas a la topología de los grafos.

Algoritmo 4 Algoritmo VF2 (Isomorfismo de subgrafos)

Entrada: Un estado intermedio s, un estado inicial s_0 y un conjunto $M(s_0) = 0$

si $M(s)$ cubre a todos los nodos de G_2 **entonces**
  devolver $M(s)$
si no
  Calcular el conjunto $P(s)$ de todos los candidatos para ser incluidos en $M(s)$
  para todo $(v, w) \in P(s)$ **hacer**
    si reglas de factibilidad son correctas para que (v, w) este incluido en $M(s)$ **entonces**
      Calcular el estado s' obtenido al adicionar (v, w) en $M(s)$
      Ejecutar este algoritmo pero para el estado s' como estado intermedio de entrada.
    fin si
  fin para
  restaurar las estructuras de datos
fin si

El algoritmo VF2 busca correspondencias entre dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$, a través de la determinación de un mapeo M en el cual están asociados los nodos de G_1 con los nodos de G_2 teniendo en cuenta algunas restricciones predefinidas. Generalmente, el mapeo M está expresado como el conjunto de pares (v, w) que representan el mapeo de un nodo $v \in G_1$ con un nodo $w \in G_2$. Por lo tanto, puede afirmarse que: el mapeo $M \subseteq V_1 \times V_2$ es un isomorfismo de grafos si y solo sí M es una función biyectiva que preserva la estructura de ramas de los dos grafos; y es un isomorfismo de sub-grafos en el caso de que M sea un isomorfismo entre G_2 y un subgrafo de G_1.

La función de mapeo se puede describir por medio de una representación estado-espacial (SSR)[3], en la cual cada estado s del proceso de correspondencia puede asociarse a una solución de mapeo parcial $M(s)$ que contiene solo un subconjunto de M. Dicho mapeo parcial $M(s)$ puede identificar unívocamente dos sub-grafos $G_1(s)$ y $G_2(s)$ obtenidos a

[3] 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 diferenciales de primer orden.

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 lo anterior, 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 (v, w)de nodos concurrentes. Por otra parte, considerando todos los posibles estados SSR, solamente un pequeño subconjunto puede ser consistente con el tipo de estructura requerida puesto que no hay condiciones que limiten la posibilidad de alcanzar una solución completa. La consistencia de la condición puede verificarse, en caso de isomorfismo de sub-grafos, cuando los grafos parciales $G_1(s)$ y $G_2(s)$ asociados a $M(s)$ son isomorfos.

De esta manera, en el presente trabajo científico, la detección de patrones fue realizada a través de un algoritmo de isomorfismo de grafos basado en el algoritmo VF2, denominado *GraphBlast*. GraphBlast fue inicialmente desarrollado para el campo de la bioinformática por Ferro y cols(Ferro y cols., 2007) y en el presente trabajo fue adaptado al contexto de los BP, *GraphBlast* con el fin de encontrar los patrones de flujo de control representados por sub-estructuras dentro de los grafos TD de cada uno de los BP del repositorio. En este caso, cada grafo TD corresponde a una *"lista de nodos y aristas"* (LNE por sus siglas en inglés) (Giugno y Shasha, 2002) basada en nodos y caminos; en la cual los nodos están etiquetados con un número (*id-nodo*) y una etiqueta (*etiqueta-nodo*), y los caminos con una lista de números id-nodo (conocida como id-camino), y una lista de etiquetas etiqueta-nodo (conocida como *etiqueta-camino*) con aristas sin etiquetas entre cada par de nodos consecutivos (figura 3.24). De este modo, *GraphBlast* genera un índice buscando todos los caminos que inician en un nodo específico y tienen una longitud predeterminada. Por ejemplo, una "*etiqueta-camino*" es: $EANDS$, y un "*id-camino*" sería: $(0, 1)$ (figura 3.25(b)).

En este caso para cada grafo y sus nodos correspondientes, el algoritmo determina todos los trayectos que empiezan en un nodo específico y tienen una longitud desde uno hasta un tamaño predefinido por una variable denominada longitud de camino (LP), la cual tiene un valor por defecto de 4 ($LP = 4$). La construcción del índice es realizada a través de un conjunto de listas "*id-camino*" e "*id-etiqueta*" utilizando una tabla de llaves (*hashtable*), las cuales representan los valores de la lista "*etiqueta-camino*". En los BP, la lista "*etiqueta-camino*" permite definir y describir un grafo de un patrón encontrado en un grafo TD. Por ejemplo, en la figura 3.25(c) 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, los índices son construidos teniendo en cuenta el número de ocurrencias de esta etiqueta-camino al interior de cada grafo almacenado.

La tabla de llaves está definida como una *"huella digital"* de la base de datos y compuesta por una matriz donde las filas son 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 es almacena en la base de datos Berkeley (Oracle, 2010). La figura 3.25(a) muestra una base de datos compuesta por tres grafos TD, cada grafo contiene una lista *id-camino* con todos los trayectos que representan una secuencia de etiquetas en una

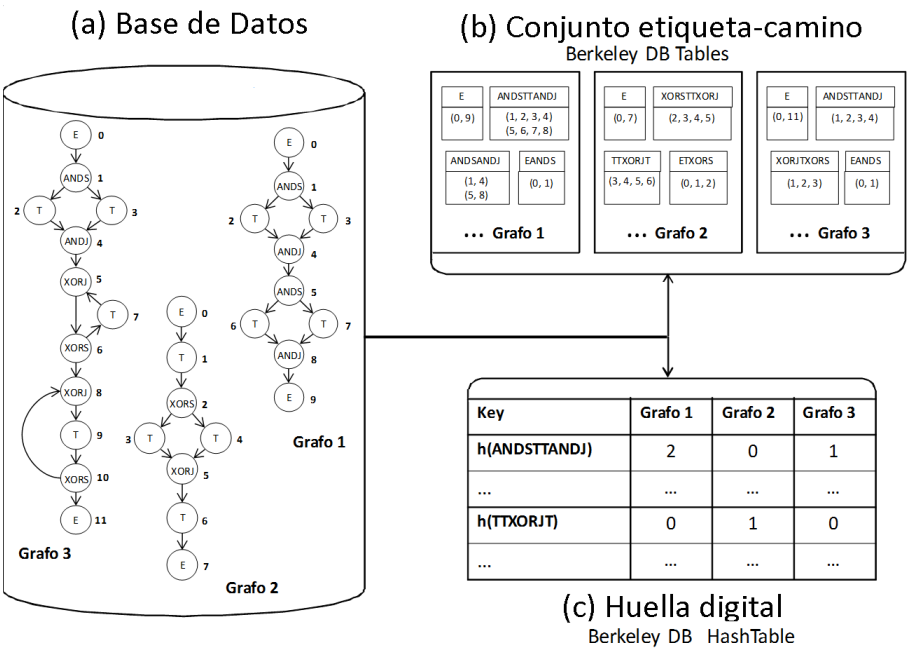


Figura 3.25: Almacenamiento de los grafos TD y la creación del índice sobre las tablas de la base de datos Berkeley. Fuente propia.

lista *etiqueta-camino* (figura 3.25(b)); y un índice que contiene la clave (*key*) localizada en la *"huella digital"*.

Organizador semántico de resultados

Esta sub-capa organiza los resultados obtenidos en la sub-capa de recuperación de BP del repositorio de acuerdo con 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 del repositorio (P_T) y el conjunto de patrones de un BP de consulta (P_Q).

$$Dp = \frac{|P_Q - P_T|}{P_Q + P_T|} \tag{3.1}$$

- **Distancia por número de nodos (Dn):** evalúa la distancia de acuerdo a la relación entre el número de nodos tipo conector (*ANDS, ANDJ, XORJ, XORS, ORS, ORJ*) encontrados en un BP del repositorio (N_t) y los encontrados en un BP de consulta (N_q).

$$Dn = \frac{|N_q - N_t|}{N_q + N_t|} \tag{3.2}$$

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

$$De = \frac{|E_q - E_t|}{E_q + E_t|} \tag{3.3}$$

- **Distancia por descriptor de ruta ($\mathbf{Drd_T}$):** evalúa la distancia en términos del número de ocurrencias y posiciones de los patrones detectados en el BP del repositorio (oN_t y PP_t) y los detectados en un BP de consulta (oN_q y PP_q). En este sentido la distancia total por descriptor de ruta (Drd_T) corresponde a la suma de todas las distancias por descriptor de ruta por cada patrón (Drd_i) detectado en los dos BP.

$$Drd_T = \sum \frac{\frac{|PP_q - PP_t|}{PP_q + PP_t}}{|oN_q - oN_t|} \tag{3.4}$$

Por ejemplo, supóngase que se tienen dos BP: BP_1 y BP_2, los cuales comparten los patrones *XORS* y *XORJ* en común. A partir de ellos se construye una matriz con las ocurrencias y posiciones de los patrones en los BP como se puede observar en la figura 3.26.

No	BP_1	BP_2
1	XORJ/6	XORS/2
2	XORS/9	XORJ/4
3		XORS/7
4		XORS/10

Posición del patrón en el BP

Figura 3.26: Ejemplo de ocurrencias y posiciones de los patrones en dos BP. Fuente propia.

En esta figura, la primera ocurrencia del patrón XORJ aparece en la posición 6 y la primera ocurrencia del patrón XORS en la posición 9 del BP_1; del mismo modo en el BP_2 la primera ocurrencia del patrón XORS es en la posición 2, la segunda en la

posición 7 y la tercera en la posición 10; y la primera ocurrencia del patrón XORJ es en la posición 4. De esta manera, la distancia Drd_i para el primer patrón (*XORJ*) se puede calcular como:

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

Y para el segundo patrón (*XORS*) como:

$$Drd_2 = \frac{\frac{|9-2|}{9+2}}{|2-1|} = \frac{7}{11}$$

Finalmente, la distancia total de descriptor de ruta es:

$$Drd_T = \frac{DDR_1 + DDR_2}{numeroDDRs} = \frac{\frac{1}{5} + \frac{7}{11}}{2} = 0{,}418$$

- **Distancia semántica de patrones (Dsp):** esta distancia evalúa la diferencia entre dos patrones de acuerdo a relaciones generales establecidas por Rusell y cols., (Russell y cols., 2006). Para esto, en el trabajo científico presentado en este libro, se creó una abstracción de relaciones entre doce patrones la cual se denominó *ontología de patrones de flujo de control* cuyas relaciones son básicamente dos: especialización y composición.

  La relación de especialización entre patrones existe cuando un patrón es una forma más restringida de otro. Por ejemplo, el patrón *"selección múltiple"* es una forma más especializada del patrón *"división paralela"*. En otras palabras el patrón *"división paralela"* presenta una funcionalidad más general correspondiente a una compuerta *AND* con comportamiento *SPLIT*, lo cual indica que el flujo de control diverge de manera síncrona.

  La relación de composición existe cuando un patrón puede expresarse a través de la combinación de dos (o más) patrones. Por ejemplo, el patrón *"selección múltiple"* tiene una funcionalidad *OR* con comportamiento *SPLIT* (flujo de control síncrono o asíncrono y divergente), que puede expresarse a través de una combinación de funcionalidades del patrón *"división paralela"* (flujo de control síncrono y divergente) y el patrón *"selección exclusiva"* (flujo de control asíncrono y divergente).

  Teniendo en cuanta las relaciones anteriores es posible encontrar la distancia semántica entre patrones contenidos en los BP utilizando la ontología de patrones y una distancia de salto (D_{salto}) (Ge y Qiu, 2008) entre dos patrones en la ontología.

$$D_{salto} = 1 + \frac{1}{2^{profundidad}} \tag{3.5}$$

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 (Dsp) es la suma de todos los valores sobre cada camino entre un par de conceptos (Algoritmo 5):

$$Dsp = \sum D_{salto} \tag{3.6}$$

Algoritmo 5 Algoritmo para función distancia semántica

Entrada: Conceptos C_q y C_t
Salida: Distancia semántica D_sp

1: D_sp = 0
2: **si** $C_q \neq C_t$ **entonces**
3: saltos = 0
4: $SC_q[]$ = obtenerSuperConceptos(C_q)
5: $SC_t[]$ = obtenerSuperConceptos(C_t)
6: CC = obtenerConceptoComun($SC_q[]$, $SC_t[]$)
7: d_1 = 0
8: d_2 = 0
9: **para** j=1 hasta j = indice(CC, SC_q) **hacer**
10: $d_1 = d_1 + 1 + \frac{1}{2^{SC_q - j}}$
11: **fin para**
12: **para** j=1 hasta j = indice(CC, SC_t) **hacer**
13: $d_2 = d_2 + 1 + \frac{1}{2^{SC_t - j}}$
14: **fin para**
15: $Dsp = d_1 + d_2$
16: **devolver** Dsp
17: **fin si**

En la presente investigación científica, la distancia semántica fue calculada por medio del algoritmo 5 . Este 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) (línea 2); ya que si esta condición no se cumple significa que los dos enriquecimientos corresponden exactamente al mismo concepto y en este caso la distancia entre ellos toma el valor de 0. Por el contrario, si los conceptos C_Q y C_T son distintos, entonces se procede a calcular los arreglos de sus superconceptos (denotados por SC_Q y SC_T respectivamente) (líneas 4 y 5). A continuación, el algoritmo encuentra un concepto en común CC entre estos dos arreglos, con el fin de definir el trayecto

más corto entre C_Q y C_T (es decir, el menor número de saltos); y 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) (líneas 6 - 14). Entonces, suma estos dos valores para obtener la distancia semántica total (Dsp) entre los dos conceptos (líneas 15 y 16).

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

$$DpT = Dp + Dn + De + Drd_T + Dsp \tag{3.7}$$

En este sentido, es posible clasificar los BP del repositorio de dos maneras: generando cinco rankings por cada una de las distancias presentadas anteriormente o alternativamente un solo ranking basado en la distancia total (DpT) empezando por aquellos con la menor DpT hasta aquellos con la mayor DpT respecto al BP de consulta. Finalmente el mecanismo de ranking seleccionado pasa al módulo de correspondencia estructural y semántica el cual ejecuta una comparación estructural exhaustiva de cada BP del repositorio (uno a uno) con el BP de consulta, con el fin de refinar el ranking inicial.

A continuación, los algoritmos 6 y 7 resumen las fases de almacenamiento y recuperación del repositorio . El algoritmo 6 recibe como entrada un documento BPMO y procede a transformarlo en las dos notaciones de grafos, previamente explicadas (grafo de proceso pg y grafo TD tdg); luego, detecta el conjunto de patrones, lo etiqueta indicando la ocurrencia de cada patrón detectado y lo almacena.

Algoritmo 6 Algoritmo para almacenar procesos BPMO

Entrada: $DocumentoBPMO, conjuntoPatrones$

1: pg = transformarGrafo(DocumentoBPMO) (*Algoritmo 1*)
2: tdg = transofrmarGrafoTD(pg) (*Algoritmo 3*)
3: **para todo** $p \in conjuntoPatrones$ **hacer**
4: buscar$p$ en tdg utilizando VF2 (*Algoritmo 4*)
5: **si** $p \subseteq tdg$ **entonces**
6: adicionar p al conjunto $patronesDetectados$
7: **fin si**
8: **fin para**
9: p' = Etiquetar p con el conjunto $patronesDetectados$
10: Almacenar p'
11: Almacenar *DocumentoBPMO*

En esta fase, el algoritmo de recuperación y generación del ranking de procesos BPMO (algoritmo 7) es el último en ejecutarse. De manera similar al algoritmo 6, este algoritmo inicia transformando un BP de consulta en las dos notaciones de grafos (grafo de proceso y grafo TD), luego detecta el conjunto de patrones que éste contiene, pero, en lugar de almacenarlo, procede a buscar un grupo de BP del repositorio que contenga un conjunto similar de patrones. Finalmente, el algoritmo organiza el conjunto de BP con patrones similares en un ranking de acuerdo a la distancia total de patrones D_{pT}.

Algoritmo 7 Algoritmo Para recuperar y generar el ranking de procesos BPMO

Entrada: $DocumentoBPMOdeConsulta, conjuntoPatrones, D_p, D_n, D_e, D_{rd_T}, D_{sp}$
Salida: $RankingBPMO$

1: pg = transformarGrafo(DocumentoBPMO) (*Algoritmo 1*)
2: tdg = transofrmarGrafoTD(pg) (*Algoritmo 3*)
3: **para todo** $p \in conjuntoPatrones$ **hacer**
4: buscar$p$ en tdg utilizando VF2 (*Algoritmo 4*)
5: **si** $p \subseteq tdg$ **entonces**
6: adicionar p al conjunto $patronesDetectados$
7: **fin si**
8: **fin para**
9: **si** $patronesDetectados \neq \emptyset$ **entonces**
10: $BPsimilares = encontrarBP(patronesDetectados, tdg)$
11: **fin si**
12: **para todo** $BPrepositorio \in BPsimilares$ **hacer**
13: $D_{pT} = D_p + D_n + D_e + D_{rd_T} + D_{sp}$
14: Adicionar $BPrepositorio$ a $RankingBPMOg$ de acuerdo a D_{pT}
15: **fin para**
16: **devolver** $RankingBPMO$

3.2.3. Capa de almacenamiento

La capa de almacenamiento está compuesta por tres repositorios (figura 3.27), en los cuales están almacenados los modelos de BP representados como: documentos BPMO, documentos de grafos TD (documentos .dat), y referencias de BP (base de datos relacional que permiten discriminar los BP del repositorio y los BP de consulta).

Modelos BP	Grafos TD			Referencias BP
Archivos WSML (Procesos BPMO)	*BP Repositorio*	*BP Consulta*	*Patrones de Control de Flujo*	*RDBMS*

Figura 3.27: Contenido de los repositorios de la capa de almacenamiento. Fuente propia.

El primer repositorio llamado *"Modelos BP"*, está encargado de almacenar los procesos BPMO en su forma original (documentos BPMO) a través del formato WSML (archivos con extensión .wsdl). El segundo repositorio, denominado *"Grafos TD"*, almacena los grafos TD (archivos con extensión .dat), para los BP del repositorio, los BP de consulta, y los patrones de flujo de control, en la base de datos de grafos *"Berkeley"* (Oracle, 2010). El último repositorio, denominado *"Referencias BP"* corresponde a una base de datos relacional que

guarda la ruta de ubicación de los documentos WSML de los BP, la ruta del respectivo grafo TD, y el conjunto de patrones detectados en él. La figura 3.28 permite observar la abstracción del modelo entidad relación correspondiente a la base de datos relacional del repositorio *referencias BP*, la cual está compuesta por cuatro tablas: *Processmodel* para almacenar las rutas (*"path"*) de la ubicación de los modelos BPMO y del grafo TD; *Patternrefference* que almacena las rutas de los doce patrones de flujo de control considerados; *Patternmodel* con el fin de almacenar información de cada patrón; y *PatternStock* para almacenar las ocurrencias de patrones (*pattern-quantity*) detectados en los modelos BPMO de la tabla *Processmodel*.

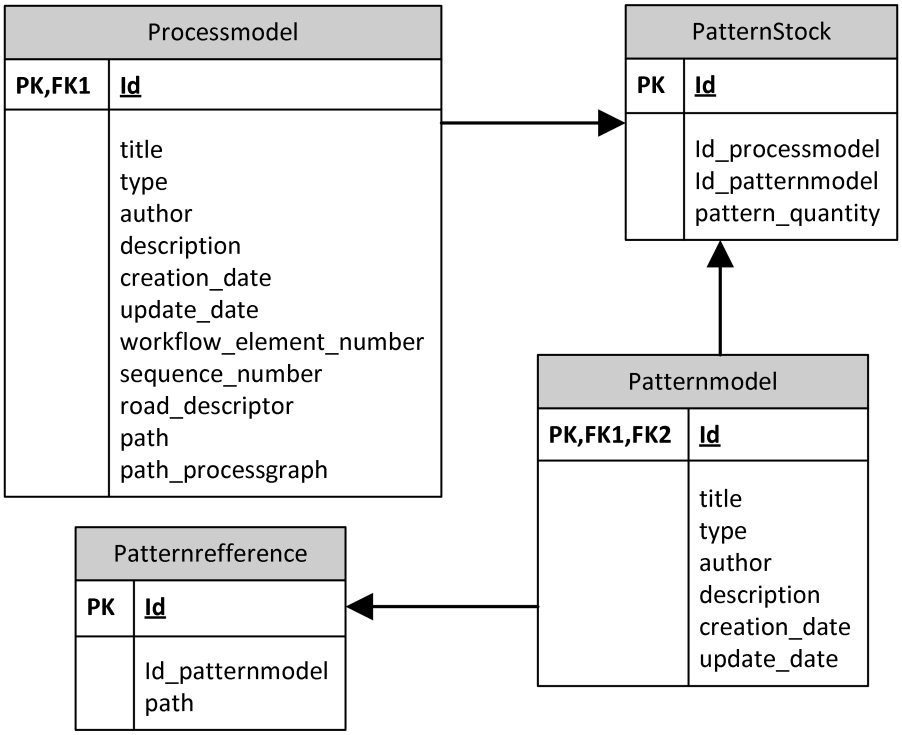


Figura 3.28: Abstracción del diagrama entidad-relación de la base de datos de referencias. Fuente propia.

3.3. Resumen

Este capítulo presentó el módulo de pre-correspondencia constituido por un repositorio de BP basado en semántica de comportamiento. El repositorio permitió almacenar y recuperar una lista ordenada (ranking) de BP de acuerdo a un conjunto de patrones de flujo de control encontrados en un BP de consulta. En la generación del ranking fueron consideradas las relaciones semánticas que existen en los patrones las cuales son establecidas a través de una ontología de patrones de flujo de control.

Capítulo 4

Correspondencia de procesos de negocio

El presente capítulo describe el mecanismo de correspondencia de BP implementado en el módulo analizador estructural y semántico (Figura 3.1(d)), el cual refina el ranking obtenido por el repositorio a través de la generación de correspondencias entre un BP de consulta y cada uno de los BP del ranking (es decir aquellos grafos que son obtenidos como resultado de la clasificación realizada por el repositorio basado en semántica del comportamiento, que en adelante serán denominados BP repositorio).

La generación de correspondencias es obtenida ejecutando una serie de operaciones de edición que modifican a cada uno de los BP del repositorio con el fin de hacerlos tan similares como sea posible al BP de consulta. Las operaciones de edición alteran propiedades de los BP como: estructura (adición y eliminación de nodos o aristas), lingüística (diferencias léxicas y semánticas entre los conceptos asociados a las interfaces de las tareas), y comportamiento secuencial (número de secuencias similares). Cada operación supone un costo de edición, de tal forma que si un BP específico del repositorio requiere más operaciones que otros BP, significa que su diferencia es mayor respecto al BP de consulta.

Como resultado de la aplicación de las operaciones de edición es generado un *"grafo editado"* por cada uno de los BP del repositorio. Este grafo, refleja los cambios realizados por la acción de las operaciones ejecutadas, y además permite obtener listas de correspondencias entre el BP de consulta y cada uno BP del repositorio.

Por ejemplo, considérense los dos BP mostrados en la figura 4.1. El primero (a) es el BP de consulta y el segundo (b) es uno de los BP del repositorio. Después de aplicar las operaciones de edición sobre el BP del repositorio se obtiene el grafo editado y el mapeo representado por líneas que interconectan a los dos BP (figura 4.2).

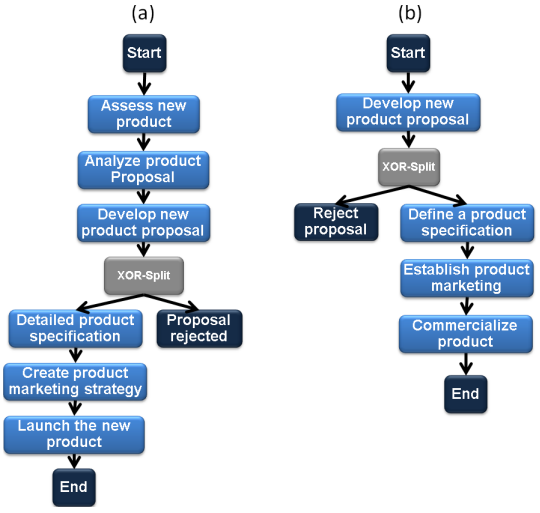


Figura 4.1: Ejemplos de dos procesos a ser comparados. Fuente propia.

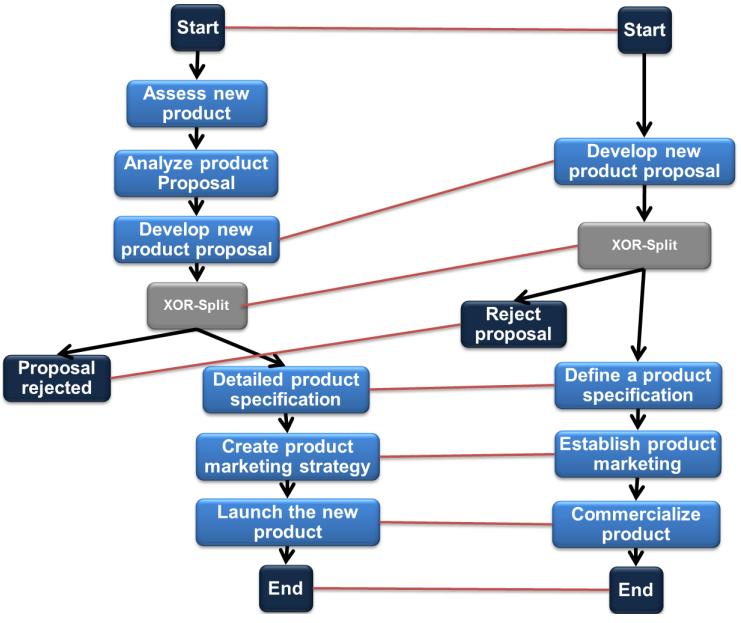


Figura 4.2: Correspondencia entre los procesos comparados en la figura 4.1. Fuente propia.

En la presente investigación, el analizador semántico y estructural implementa una arquitectura base compuesta por cinco módulos (figura 4.3): un analizador estructural (a), un analizador lingüístico de tareas (b), funciones de costo (c), funciones de similitud (d), y un organizador estructural de resultados (e). Estos módulos están descritos a continuación.

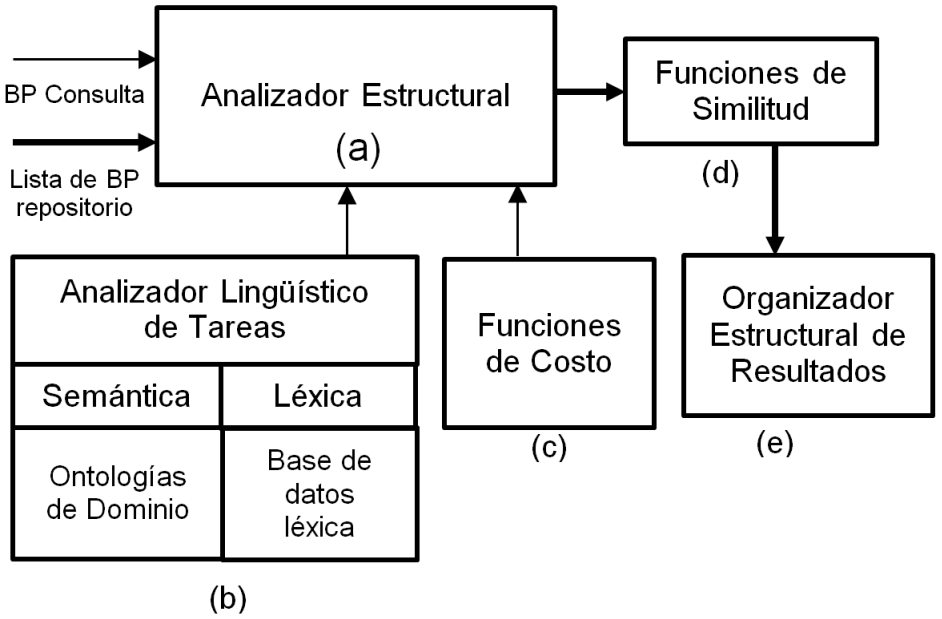


Figura 4.3: Analizador semántico y estructural. Fuente propia.

4.1. Analizador de correspondencia estructural

El analizador estructural (Figura 4.3(a)) recibe como entrada un BP de consulta y lo compara estructuralmente con cada uno de los BP del ranking obtenido por el repositorio. La correspondencia estructural, ejecutada en este módulo, está basada en el algoritmo A* para isomorfismo de sub-grafos por corrección de errores (Messmer, 1995) previamente adaptado por Corrales (Corrales, 2008). Este algoritmo empieza con la creación de un conjunto de mapeos desde un nodo de un BP de consulta a cada nodo de los BP del repositorio. Cada mapeo involucra un costo debido a la aplicación de las operaciones de edición sobre un BP del repositorio(eliminar, insertar o sustituir nodos y aristas).

De manera similar, el analizador estructural implementa un algoritmo de isomorfismo con corrección de errores (algoritmo 8) el cual crea diferentes mapeos ejecutando operaciones de edición sobre los BP del repositorio hasta encontrar un mapeo mínimo para cada BP (un mapeo con el mínimo costo de edición) o cuando todos los mapeos posibles han excedido un valor de tolerancia predefinido por el usuario (AC - costo aceptable).

En la presente investigación, los grafos de proceso (BP de consulta y BP del repositorio) están compuestos por nodos de tipo tarea, los cuales representan actividades de negocio; conectores que expresan restricciones de flujo de control; y aristas que conforman enlaces entre los nodos.

Algoritmo 8 Algoritmo de correspondencia estructural y semántica

Entrada: $G_q(V_q), G_t(V_t)$
Salida: p

1: Incializa $Open$ (mapea cada nodo tarea de V_q en cada nodo tarea de V_t)
2: Crea un mapeo p
3: Calcula $C(p)$ (el costo del mapeo)
4: Agregar p a $Open$
5: **si** $Open = \emptyset$ **entonces**
6: Salir
7: **fin si**
8: Obtener $p \in Open$ tal que $C(p)$ sea mínimo
9: Remover p de $Open$
10: **si** $C(p) > AC$ **entonces**
11: FIN
12: **fin si**
13: **si** p es un mapeo completo desde G_q a G_t **entonces**
14: **devolver** p
15: $AC = C(p)$
16: ir a la línea 4
17: **fin si**
18: Sea $p = \{(v_1, w_i), ..., (w_k, w_j)\}$ el mapeo actual de k nodos de G_q
19: **para todo** $w \in V_t$ que no ha sido mapeado a un nodo en V_q **hacer**
20: extender el mapeo actual p a p' mapeando el nodo $V_{k+1} \in V_q$ a $w \in V_t$
21: $p' = \{(v_1, w_i), ..., (w_k, w_j), (w_{k+1}, w)\}$
22: calcular $C(p')$
23: **si** $(w_{k+1}, w) \in ConjuntoFunciones$ **entonces**
24: $C_{Funcion} = CorresponenciaFunciones(w_{k+1}, w)$
25: **fin si**
26: **si** $(w_{k+1}, w) \in ConjuntoEventos$ **entonces**
27: $C_{Evento} = CorresponenciaEventos(w_{k+1}, w)$
28: **fin si**
29: adicionar p' a $Open$
30: **si** $\exists e(w, w')$ **entonces**
31: $C_{arista} = CostoArista(e)$
32: **fin si**
33: **si** $\exists c(w, w')$ (un conector entre w y w') **entonces**
34: $C_{conector} = CostoConector(c)$
35: **si no**
36: $C_{elimninacion} = CostoEliminacion(v, v')$
37: **fin si**
38: Adicionar $Distancia(C_{arista}, C_{conector}, C_{elimninacion})$ a $C(p')$
39: **fin para**
40: ir a la línea 4

Por lo tanto, el conjunto de operaciones de edición válidas que el analizador estructural puede ejecutar son: sustituir o eliminar nodos de tipo tarea; insertar o eliminar conectores; e insertar y eliminar aristas. Cada operación realizada implica un costo de edición el cual representa la distancia de cada BP del repositorio respecto al BP de consulta. En la presente investigación, es posible fijar valores que asignen diferentes pesos a cada operación de edición para permitir que los usuarios personalicen las restricciones del analizador estructural y de esta manera 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.

Adicionalmente, el algoritmo de isomorfismo con corrección de errores (algoritmo 8) discrimina el tipo de nodos a comparar (función, conector, evento) con el fin de reducir el espacio de búsqueda. De esta manera crea grupos de nodos similares entre los grafos de los BP de consulta (G_Q) y del repositorio (G_T). Por esta razón la correspondencia entre los dos grafos G_Q y G_T solamente ejecuta comparaciones entre nodos que pertenecen al mismo tipo.

El algoritmo 8 inicia calculando una función denominada $open$ que contiene los mapeos entre los nodos de tipo tarea pertenecientes al conjunto V_q del BP de consulta y al conjunto V_t de un BP del repositorio. A continuación, crea un mapeo p relacionando el primer nodo incluido en el primer conjunto V_q del grafo G_Q con todos los nodos del conjunto V_t del grafo G_T, calcula su costo de edición ($C(p)$) y agrega el mapeo p a la función $open$. Repite el mismo procedimiento para cada uno de los nodos del grafo G_T hasta obtener un mapeo p' que sea mínimo (con un costo mínimo de edición).

El mapeo p' conocido como mapeo parcial, es posteriormente extendido por medio del mapeo de un nodo v del grafo G_T (v es un nodo que aún no ha sido mapeado) a un nodo w en el grafo G_Q (w es un nodo que no pertenece al mapeo actual). Esto implica la ejecución de operaciones de edición de nodos y aristas (*CostoFuncion, CostoEvento, CostoArista y CostoConector*) en las cuales, los atributos de v son sustituidos por atributos de w; y para cada par de nodos ya mapeados (v', w') debe cumplirse que cada arista (v', v) en G_Q pueda mapearse a una arista (w', w) en G_T por medio de operaciones de edición de arista. El algoritmo finaliza cuando se alcanza un estado que representa un isomorfismo de subgrafos con corrección de error de G_Q a G_T (mapeo mínimo); o cuando todos los estados en el espacio de búsqueda tienen costos de edición que exceden el costo de aceptación (AC). El costo del mapeo parcial ($C(p')$) representa el costo de extender el mapeo actual p con el siguiente nodo en el grafo consulta.

Los costos de edición son calculados con las funciones $CostoFuncion$, $CostoEvento$, $CostoArista$ y $CostoConector$; y el costo total es estimado por medio de la suma de las mismas. El módulo encargado de ejecutar dichas funciones es conocido como el módulo de funciones de costo (Figura 4.3(c)), el cual está descrito con detalle en la próxima sección.

4.2. Funciones de costo

Este módulo calcula los costos de las operaciones de edición para estimar la distancia entre un BP de consulta y cada uno de los BP del repositorio, utilizando las expresiones matemáticas y los algoritmos correspondientes descritos a continuación.

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

  En este caso el algoritmo 9 analiza si las operaciones de insertar, sustituir una arista o eliminar un conector entre (w, w') son necesarias para cada mapeo $M(v, w)$ y $M'(v', w')$ (dónde $(v, v') \in BPdeconsulta$ y $(w, w') \in BPdelrepositorio$), suponiendo que existe una arista entre los nodos (v, v').

Algoritmo 9 Algoritmo para calcular el costo de arista

Entrada: $M(v, w)$ y $M'(v', w')$, donde $(v, v') \in$ BPConsulta y $(w, w') \in$ BPRepositorio
Salida: ed

1: **para todo** $M(v, w)$ y $M'(v', w')$ **hacer**
2: **si** $\exists e(w, w')$ **entonces**
3: **devolver** $ed = 0$
4: **si no, si** $\nexists e(v, v')$ **entonces**
5: **devolver** $ed = (\vartheta * ie)$
6: **si no, si** $\exists c(w, w')$ **entonces**
7: **devolver** $ed = \{(\vartheta * ie) + (\mu * de)\}$
8: **fin si**
9: **fin para**

- **Distancia de conector (cd):** esta función cuenta el número de conectores sustituidos, insertados y eliminados en cada BP del repositorio y los multiplica por el factor de conector sustituido (ω), conector eliminado (ρ) o conector insertado (σ) predefinido por el usuario.

  En este caso el algoritmo 10, analiza si las operaciones de insertar, sustituir un conector o eliminar una arista entre (w, w') son necesarias para cada mapeo $M(v, w)$ y $M'(v', w')$ (dónde $(v, v') \in BPdeconsulta$ y $(w, w') \in BPdelrepositorio$), suponiendo que existe un conector entre los nodos (v, v'), .

- **Distancia de nodo tipo tarea (tnd):** esta función cuenta el número de nodos tipo tarea eliminados (dn) y el costo de sustituir un nodo del BP de consulta por un nodo de un BP del repositorio (sn). La operación de sustitución de nodos tipo tarea es calculada por el módulo analizador lingüístico (Figura 4.3(b)), descrito en la sección 4.3, el cual permite encontrar un nodo en el BP del repositorio que pueda reemplazar a un nodo del BP de consulta con un costo mínimo (es decir, con nombre e interfaces similares)

Algoritmo 10 Algoritmo para calcular la función $CostoConector$

Entrada: $M(v, w)$ y $M'(v', w')$, donde $(v, v') \in$ BPConsulta y $(w, w') \in$ BPRepositorio
Salida: cd

1: **para todo** $M(v, w)$ y $M'(v', w')$ **hacer**
2: **si** $\exists c(w, w')$ **entonces**
3: **si** tipo de $c(w, w') \neq$ tipo de $c(v, v')$ **entonces**
4: **devolver** $cd = \omega * sc$
5: **si no**, **si** tipo de $c(w, w')$ = tipo de $c(v, v')$ **entonces**
6: **devolver** $cd = 0$
7: **fin si**
8: **si no**, **si** $\nexists c(w, w')$ **entonces**
9: **devolver** $cd = \varphi * ic$
10: **fin si**
11: **si** $\exists e(w, w')$ **entonces**
12: **devolver** $cd = \rho \sum dc + \sigma \sum ic$
13: **fin si**
14: **fin para**

Tal como las funciones anteriores, esta función utiliza factores de multiplicación predefinidos por los usuarios como: el factor de nodo eliminado (τ) y el factor de nodo sustituido (φ).

$$tnd = \tau \sum dn + \varphi \sum sn \tag{4.1}$$

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

$$TD = ed + cd + tnd \tag{4.2}$$

4.3. Analizador lingüístico

El analizador lingüístico (Figura 4.3(b)) evalúa la operación de sustitución comparando los nodos tipo tarea de un BP de consulta y un BP del repositorio. Los nodos tipos tarea que son comparados 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 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 fue diseñado 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 los nombres de dos nodos, analizando combinaciones de palabras y abreviaciones.

En la presente investigación, para encontrar la distancia léxica, fueron utilizados los siguientes algoritmos: "*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; "*Check abbreviation*" para comparar abreviaciones; y "*Check synonym*" para encontrar sinónimos a partir de una base de datos léxica (como por ejemplo "*WordNet*" (Miller, 1995)). Adicionalmente, fue tomada la ecuación propuesta por Patil, Abhijit y cols., (Patil, Oundhakar, Sheth, y Verma, 2004), la cual recibe el resultado de los algoritmos NGram (m_1), Check Synonym (m_2) y Check abbreviation (m_3) y evalúa la distancia léxica (LS) entre los nombres de dos nodos.

$$LS = \begin{cases} 1 & \text{si } m_1 = 1 \vee m_2 = 1 \vee m_3 = 1 \\ m_2 & \text{si } (0 < m_2 < 1) \wedge (m_1 = m_3 = 0) \\ 0 & \text{si } m_1 = m_2 = m_3 = 0 \\ \frac{m_1+m_2+m_3}{3} & \text{si } m_1, m_2, m_3 \in (0,1) \end{cases} \tag{4.3}$$

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 es realizado utilizando dos ontologías de dominio: una para los identificadores (nombres de las tareas) y otra para las interfaces (entradas y salidas). El desarrollo de este módulo está basado en el trabajo de Ordóñez y Bastidas (Ordóñez, Bastidas, y Corrales, 2012) en el cual desarrollaron un mecanismo de comparación semántica entre tareas de dos BP.

A continuación, es presentado un breve estudio de las ontologías del dominio de las telecomunicaciones.

Ontologías del dominio de las telecomunicaciones

Una de las claves para lograr la interoperabilidad de los sistemas o aplicaciones es contar con un lenguaje estandarizado que permita el entendimiento común de los conceptos. En esta medida surge la necesidad de encontrar un modelo que permita obtener una visión general de los conceptos más utilizados por los operadores de telecomunicaciones, y que además sea computacionalmente comprensible (es decir que pueda ser interpretado y procesado por una aplicación o sistema software); estas características están proporcionadas por las

ontologías de dominio las cuales son necesarias para llevar a cabo el propósito de la presente investigación científica, puesto que constituyen la herramienta que facilta el enriquecimiento semántico de los parámetros que definen las funcionalidades de las tareas de los BP (descritas en términos de sus entradas, salidas e identificadores).

En este aspecto, fueron estudiadas varias propuestas de ontologías en el dominio de las telecomunicaciones (Frankowski, Rubach, y Szczekocka, 2007; Dawidziuk y Cieśĺak, 2007; Floch, 2007; Watkins y Richardson, 2005; Wahle y Magedanz, 2008; Devitt, Danev, y Matusikova, 2006; Qiao y Li, 2010), sin embargo gran parte de las ontologías presentadas en estos trabajos están limitadas para el modelado del conocimiento de un sector muy específico o de una empresa particular, por lo tanto su aplicación dentro de la presente aproximación científica no es viable porque no favorece el carácter genérico y la escalabilidad deseada.

No obstante, uno de los proyectos importantes encontrados corresponde a un marco de referencia semántico para el sector de las telecomunicaciones denominado YATOSP (Yet Another Telecommunication Ontologies, Process And Services Framework), el cual busca proporcionar un puente entre la arquitectura del proyecto SUPER y un conjunto de ontologías basadas en la iniciativa NGOSS (New Generation Operations Systems and Software) del TMF (Telemanagement Forum) denominadas SeTOM, SSID y STAM, las cuales definen BP semánticos en el dominio de telecomunicaciones mediante la extracción de información común entre las compañías de este sector (Martínez y Pérez, 2008; Belecheanu y cols., 2007). A pesar de que existe una buena documentación en cuanto a la descripción de la arquitectura de YATOSP, este trabajo carece de ejemplos de ejecución que permitan verificar la validez de sus planteamientos y el proyecto SUPER restringe el acceso a algunos documentos de gran relevancia para su posible implementación.

En la investigación científica descrita en este libro, el marco de referencia YATOSP permitió centrar la atención en la iniciativa NGOSS, especialmente en el conjunto de modelos propuestos para la gestión de los BP, la reingeniería de procesos y la integración de aplicaciones empresariales propias del sector de las telecomunicaciones: el Mapa de Operaciones de Telecomunicaciones Mejorado (eTOM, enhanced Telecom Operations Map), el Modelo de Información/Datos Compartidos (SID, Shared Information/Data Model), el Mapa de Aplicaciones de Telecomunicaciones (TAM, Telecom Applications Map) y la Arquitectura Neutral de Tecnologías (TNA, Technology Neutral Architecture) (Fleck, 2003).

Teniendo como referencia estas propuestas fue posible establecer una relación de correspondencia entre la forma en que están descritas las tareas de un BP y la información contenida en las ontologías. De esta manera, el identificador o nombre de una tarea de un BP puede modelarse a través del estándar eTOM, mientras que las entradas y salidas a través de las entidades que conforman el modelo SID. Para entender mejor estas relaciones, a continuación es presentada una descripción de las principales características de estos dos modelos (eTOM y SID) y de las ontologías desarrolladas a partir de ellos (SeTOM y SSID) por el Telemanagement Forum.

- **Mapa mejorado de operaciones de telecomunicaciones (eTOM):** eTOM es un marco de referencia de BP, que proporciona un modelo para la categorización de todas las actividades involucradas en la operación de un proveedor de servicios de telecomunicaciones. Provee una terminología común de la industria de las telecomunicaciones en cuanto a procesos e información, la cual facilita y soporta las relaciones operador-a-operador, operador-a-cliente y operador-a-socio/proveedor respondiendo con esto a la necesidad de rápido despliegue de servicios y el manejo de problemas que exige el entorno competitivo en la actualidad (ITU, 2005).

  Por esta razón, eTOM organiza en forma jerárquica y coherente los conceptos que constituyen el conocimiento relativo a los procesos, subprocesos y actividades llevadas a cabo al interior de un proveedor de servicios de telecomunicaciones. Estos procesos pueden clasificarse en tres grandes áreas que abarca el modelo eTOM: Estrategia-Infraestructura-Producto, Operaciones y Gestión Empresarial

  El área estrategia-infraestructura-producto abarca los procesos involucrados en la elaboración de estrategias a partir de los cuales la empresa planifica: el desarrollo y la gestión de la entrega y la cadena de suministro, y el mejoramiento de infraestructura y productos. En este caso entíendase por infraestructura no solo el conjunto de recursos de tecnologías de la información y de red que soportan el despliegue de productos y servicios, sino también la infraestructura operativa y organizacional necesarias para sustentar la comercialización, ventas y procesos de servicios y cadena de suministro.

  Los procesos pertenecientes a las áreas de estrategia-infraestructura-producto y Operaciones están organizados a su vez dentro de una matriz en las cual: Las divisiones verticales representan flujos de proceso extremo a extremo, que atraviesan capas horizontales de interfaces con el cliente, servicios, recursos y socios/proveedores (Rodriguez y García, 2008). En cada capa horizontal están agrupados los procesos u operaciones más específicos, y dentro de estos bloques están distribuidas las actividades en las cuales son soportados dichos procesos.

  El área de operaciones comprende aquellos procesos que proveen soporte directo al cliente, en ella pueden identificarse cuatro agrupaciones de procesos de extremo a extremo: aseguramiento, cumplimiento, facturación, y soporte y disposición de operaciones. Ésta última está encargada de asegurar que las tres agrupaciones previas puedan responder a los requerimientos del cliente preparando la información, productos, servicios y recursos, así como los proveedores y socios para entregar y sustentar las instancias de servicio del cliente.

  El área de gestión empresarial incluye los procesos que soportan directamente la administración y el funcionamiento de cualquier empresa, comprendiendo aquellos que están dirigidos hacia el establecimiento y alcance de metas y objetivos corporativos

y el soporte de servicios requeridos en la operación de toda la organización. Entre estos procesos pueden encontrarse: la gestión financiera y la gestión de recursos humanos.

En la actualidad eTOM es un estándar ampliamente empleado y aceptado para los BP en la industria de las telecomunicaciones(Burrows y Pendleburry, 2008), por lo cual ha sido utilizado dentro del marco de referencia semántico YATOSP, para formular una ontología basada en esta iniciativa de estandarización denominada SeTOM. Esta ontología, gracias al carácter de estándar de eTOM y a su amplia aceptación dentro del sector de las telecomunicaciones, se presenta como una alternativa adecuada para abordar el enriquecimiento semántico de los identificadores de las tareas de los BP de un operador de telecomunicaciones. La utilización de la ontología SeTOM está enmarcada en el enriquecimiento de procesos que están fundamentalmente relacionados con el desarrollo, soporte y despliegue de servicios de telecomunicaciones.

- **Modelo de datos/información compartido (SID):** SID es un modelo unificado de referencia de datos que proporciona un conjunto único de términos, conexiones y relaciones comúnmente utilizados en los BP básicos del dominio de las telecomunicaciones (TM-Forum, 2005). Este modelo pretende facilitar la comunicación entre los profesionales de las áreas de negocio, los desarrolladores de software, y otros empleados de las empresas, promoviendo la utilización de estos términos para referirse o describir la misma información en los BP.

  Para esto, SID captura los conceptos y principios necesarios en la definición de un modelo de información compartida, y define en detalle muchos elementos del negocio (“Entidades”) y sus atributos asociados. Una entidad de negocio es un objeto de interés en las empresas tales como cliente, producto, servicios o red; mientras que sus atributos son características que describen a la entidad. Estos términos en conjunto proporcionan una perspectiva orientada a la información y los datos del negocio que son necesarios para el funcionamiento de una empresa (Lambea, 2009).

  La arquitectura de SID está diseñada como un modelo en capas o niveles, las cuales dividen los datos/información en ocho dominios: mercado/ventas, producto, cliente, servicio, recurso, suministrador/socio, empresa y entidades de negocio comunes. En la capa más superficial, cada uno de los ocho dominios de información está alineado con el marco de referencia eTOM. En este modelo existe un un acoplamiento débil (flexible) entre los dominios y un alto grado de cohesión entre las entidades de cada dominio, lo cual facilita la segmentación del problema de negocio en piezas manejables, a la vez que permite que los recursos estén concentrados en un área particular de interés: en otras palabras, para un BP particular que esté siendo automatizando, SID permite identificar la información necesaria dentro de su arquitectura de soporte.

  Adicionalmente, SID contiene las definiciones de las entidades empresariales que participan en la gestión de las operaciones de telecomunicaciones, y permite la

integración y la interoperabilidad de los modelos de la iniciativa NGOSS, dado que corresponde al medio por el cual la información útil (es decir, la que información relevante para un determinado conjunto de BP) puede ser compartida y empleada en múltiples BP dentro de todo el sistema (TM-Forum, 2002).

Una de las ventajas que tiene la utilización de SID es que proporciona una fusión de la terminología de las empresas de telecomunicaciones en un modelo simple de información del negocio, haciendo posible el desarrollo de aplicaciones OSS/BSS (operational/business support systems) con interfaces interoperables. De esta forma SID constituye un marco de referencia para la representación de información que pueden ser compartida y reutilizada por aplicaciones OSS/BSS de múltiples proveedores (Rodriguez y García, 2008).

La ontología SSID propuesta en YATOSP, presenta más de mil conceptos organizados en diez niveles de jerarquía, que conforman un vocabulario común ampliamente utilizado dentro del sector de las telecomunicaciones, los cuales cubren términos como cliente, producto, servicio, recurso, proveedor, entre otros (Martínez y Pérez, 2008). Por esta razón, es recomendable la utilización de esta ontología dentro de la presente investigación científica, puesto que sus conceptos están estrechamente relacionados con la información compartida tanto entre aplicaciones informáticas como entre el personal al interior de un operador de telecomunicaciones. De esta manera es posible realizar el enriquecimiento semántico de las entradas y salidas de las tareas de BP con conceptos pertenecientes a esta ontología, dado que estos corresponden a la información intercambiada para realizar las distintas actividades del negocio.

En resumen, en la presente aproximación científica son utilizadas dos ontologías del dominio de las telecomunicaciones *SeTOM*(Shangguan, Gao, y Zhu, 2008; ITU, 2005) y *SSID* (TM-Forum, 2005). La ontología SeTOM útil para enriquecer semánticamente los identificadores de las tareas y la ontología SSID para las interfaces. De esta manera el cálculo de la similitud semántica puede realizarse a través de la distancia presentada en la sección de clasificación semántica de BP (sección 3.2.2) del repositorio. La distancia semántica puede ser evaluada a partir de la distancia de salto (D_{salto}) entre dos conceptos, la cual es calculada a través del razonador WSML2reasoner (Grimm, Keller, Lausen, y Nagypál, 2007). A continuación se describe el cálculo de la similitud semántica aplicada tanto a los identificadores como a las interfaces.

Similitud semántica de identificadores

Este análisis fue realizado teniendo en cuenta los niveles 2 y 3 del modelo eTOM en los cuales están definidas 5 categorías de correspondencia entre los identificadores.

- **Exacto:** 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 tres condiciones. Primero que el concepto del enriquecimiento de la tarea de consulta debe pertenecer al nivel 3 del modelo eTOM. Segundo, que el concepto del enriquecimiento de la tarea publicada debe pertenecer al nivel 2 del modelo eTOM. Y tercero, que 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.

- **Subsunción:** para clasificar el grado de correspondencia en esta categoría es necesario cumplir tres condiciones. Primero, que el enriquecimiento de la tarea publicada debe pertenecer al nivel 3 de eTOM. Segundo, que el enriquecimiento de la tarea de consulta debe pertenecer al nivel 2 de eTOM. Y tercero, que el enriquecimiento de la tarea de consulta debe hacer parte del conjunto de componentes que conforman el enriquecimiento de la tarea publicada.

- **Intersección:** 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.

- **Fallo:** cuando ninguna de las anteriores condiciones sea cumplida.

Las anteriores categorías fueron utilizadas en la ejecución del algoritmo 11, el cual calcula el grado de correspondencia entre parejas de identificadores de las tareas comparadas.

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 *exacto*. De lo contrario, el algoritmo procede a encontrar el nivel al que pertenecen dichos conceptos (LI_q y LI_t) y verifica que tanto I_q como IT pertenezcan a los niveles 2 o 3 de la misma área funcional. Si es así, entonces 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 es establecida como *intersección*. Si no es así, 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 *subsunción*. Finalmente, si ninguna de las condiciones anteriores se cumple, entonces puede afirmarse que el grado de correspondencia es *fallo*.

En el algoritmo 11 las notaciones SLC_q y SLC_t corresponden a los conceptos de segundo nivel de las tareas de consulta, y del repositorio respectivamente; y LI es el nivel de los conceptos en la ontología.

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

$$SS = \frac{1}{(p * SD + 1)} \tag{4.4}$$

Algoritmo 11 Algoritmo para determinar el grado de correspondencia entre parejas de conceptos

Entrada: I_q, I_t
Salida: $GradoCorrespondencia$

si $I_q = I_t$ **entonces**
  devolver $GradoCorrespondencia = Exacto$
si no
  $LI_q = obtenerNivel(I_q)$
  $LI_t = obtenerNivel(I_t)$
  si $(LI_q \neq 2 \vee LI_q \neq 3) \vee (LI_t \neq 2 \vee LI_t \neq 3)$ **entonces**
    devolver $GradoCorrespondencia = Fallo$
  si no
    $SLC_q = obtenerConceptoSegundoNivel(I_q)$
    $SLC_t = obtenerConceptoSegundoNivel(I_t)$
    si $SLC_q = SLC_t$ **entonces**
      si $LI_q = 3 \wedge LI_t = 3$ **entonces**
        devolver $GradoCorrespondencia = Interseccion$
      si no, si $LI_t = 2$ **entonces**
        devolver $GradoCorrespondencia = Plugin$
      si no
        devolver $GradoCorrespondencia = Subsuncion$
      fin si
    si no
      devolver $GradoCorrespondencia = Fallo$
    fin si
  fin si
fin si

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 asignan de la siguiente manera: si el grado de correspondencia es exacto $p = 0$, si es plugin $p = 0{,}2$, si es subsunción $p = 0{,}6$, si es intersección $p = 0{,}8$ y si es fail $p = 1$.

Para la determinación de la similitud semántica entre los identificadores de dos tareas (del BP de consulta y del BP del repositorio) fue empleado el algoritmo 12, el cual inicia evaluando el grado de correspondencia. De manera tal que si es $exact$ entonces ($p = 0$) entonces el valor de la similitud semántica (SS) es 1; de lo contrario, se calcula la distancia semántica (SD) y el grado de correspondencia (MD) entre C_Q y C_T. Luego, de acuerdo con el resultado obtenido para MD y dependiendo de la ontología registrada en DO (En la presente investigación se utilizó la ontología $SSID$) se ajusta el parámetro p que interviene en el cálculo de la similitud semántica junto con SD.

Algoritmo 12 Algoritmo para calcular similitud semántica entre conceptos

Entrada: C_q, C_t
Salida: SS (Similitud Semántica)

$p = 0, SD = 0$
$MD = encontrarGradoCorrespondencia(C_q, C_t)$ (Algoritmo 11)
si $MD \neq Exacto$ **entonces**
  $SD = encontrarDistanciaSemantica(C_q, C_t)$
  si MD = Plugin **entonces**
    si $DO = SSID$ **entonces**
      $p = 0{,}7$
    si no
      $p = 0{,}2$
    fin si
  si no, **si** MD = Subsuncion **entonces**
    si $DO = SSID$ **entonces**
      $p = 0{,}9$
    si no
      $p = 0{,}5$
    fin si
  si no, **si** MD = Interseccion **entonces**
    si $DO = SSID$ **entonces**
      $p = 0{,}9$
    si no
      $p = 0{,}1$
    fin si
  si no
    $p = 1$
  fin si
fin si
devolver $SS = \frac{1}{(p*SD)+1}$

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 la investigación descrita en este libro, la comparación de entradas/salidas está basada en el trabajo de Benatallah, Hacid y cols., (Benatallah y cols., 2003) en el cual a partir de una petición de servicio son encuentradas 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 del repositorio (T_T) más similar a una tarea del BP de 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

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 está descrito a continuación para el caso de las entradas considerando que para las salidas es aplicado el mismo procedimiento.

- **Similitud semántica de entradas:** la similitud entre parejas de entradas de una T_T y una T_Q, es calculada a partir del mecanismo para la comparación de conceptos explicado en el algoritmo 12. Posteriormente los resultados son organizados en una matriz de similitud, cuyas filas corresponden a las entradas de la TQ y las columnas a las entradas de la TT. Dicha matriz es analizada 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 S es definida 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 el conjunto de similitudes máximas determinado como la agrupación de similitudes S_{ij} que cumplen con la siguiente propiedad:

$$S_{ij} : (\forall S_{kj}; k = (0, 1, 2, ..., m) \Rightarrow S_{ij} > S_{kj}) \wedge (\forall S_{il} l = (0, 1, 2, ..., n) \Rightarrow S_{ij} > S_{il}) \tag{4.5}$$

  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 son presentadas 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, la similitud semántica entre entradas o salidas (SS_{IO}) es definida como:

$$SS_{IO} = \frac{\sum S_{ij}}{min(m, n) + \beta * |m - n|} \tag{4.6}$$

  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, o en el caso contrario valor de 1.

A partir de este mecanismo está definido el algoritmo que puede calcular la similitud semántica de entradas (Algoritmo 13). Este algoritmo recibe como parámetros las dos tareas a comparar (T_Q, T_T).

Algoritmo 13 Algoritmo para calcular similitud semántica entre entradas de una tarea

Entrada: T_q, T_t (Tareas de los BP)
Salida: SS_{IO} (Similitud Semántica)

1: $I_q[m] = obtenerEntradas(T_q)$
2: $m = cardinalidad(I_q)$
3: $I_t[n] = obtenerEntradas(T_t)$
4: $n = cardinalidad(I_t)$
5: $IE_q[m] = obtenerEriquecimientoEntradas(T_q)$
6: $IE_t[m] = obtenerEriquecimientoEntradas(T_t)$
7: **si** $I_q[m] = \emptyset \wedge I_t[n] = \emptyset$ **entonces**
8: **devolver** $SS_{IO} = 1$
9: **si no, si** $I_q[m] = \emptyset \vee I_t[n] = \emptyset$ **entonces**
10: **devolver** $SS_{IO} = 0$
11: **si no**
12: $ISM[m][n] = obtenerMatrizSimEntradas(IE_q[m], IE_t[n])$
13: $MSA[l] = obtenerArregloMaxSim(ISM[m][n])$
14: $MSS = adicionarArregloMaxSim(MSA[l])$
15: **si** $m > n$ **entonces**
16: **devolver** $SS_{IO} = \frac{MSS}{n+(0{,}8*|m-n|)}$
17: **si no**
18: **devolver** $SS_{IO} = \frac{MSS}{m+|m-n|}$
19: **fin si**
20: **fin si**

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 (líneas 1-2). De la misma manera extrae las entradas de T_T para almacenarlas en $I_T[]$ y guarda el tamaño de este arreglo en n (líneas 2-3). Posteriormente obtiene el enriquecimiento de las entradas de T_Q en el arreglo $IE_Q[m]$, y de la T_T en el arreglo $IE_T[n]$ (líneas 5-6).

Con el fin de verificar si existen entradas a comparar, el algoritmo evalúa si los arreglos $I_Q[]$ e $I_T[]$ están vacíos, es decir si ninguna de las tareas comparadas tiene entradas, con lo cual establece que la similitud de entradas (SS_{IO}) es igual a 1. De lo contrario, examina si alguno de ellos está vacío (si cualquiera de las dos tareas no tiene entradas), con lo cual SS_{IO} asigna el valor de 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 (líneas 7-10). Si ninguna de las anteriores condiciones se presenta, 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 $IE_T[n]$,) (líneas 11-14). Luego, analiza esta matriz para obtener las máximas similitudes ($MSA[]$). Una vez reunidos estos resultados, evalúa si la T_Q tiene más entradas que la T_T, en cuyo caso calcula la similitud semántica de entradas a partir de la ecuación 4.6 (líneas 15-19).

- **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, el algoritmo debe reemplazar los arreglos de entradas $I_T[]$, $I_Q[]$, $IE_Q[m]$, $IE_T[n]$, y 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, el algoritmo procede a calcular la similitud semántica general (OSS) entre las dos tareas comparadas ejecutando una suma ponderada de los resultados (Ecuación 4.7):

$$OSS = (SS * wId) + (SS_I * wIn) + (SS_O * wO) \tag{4.7}$$

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; 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; 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 proporcionados por los usuarios.

Finalmente, la distancia de nodo tipo tarea puede calcularse a través el algoritmo 14, el cual a su vez utiliza los algoritmos descritos para la similitud semántica de identificadores (Algoritmo 12) y la similitud semántica de interfaces (Algoritmo 13).

4.4. Resultados de similitud

Este módulo (Figura 4.3(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.

Algoritmo 14 Algoritmo para calcular la función $CostoTarea$

Entrada: T_q, T_t
Salida: C_{tarea}
1: **si** $tipodeT_q \neq tipodeT_t$ **entonces**
2: **devolver** $C_{tarea} = 1$
3: **si no**, **si** $T_q \in TipoFuncion$ **entonces**
4: **si** T_q y T_t tienen enriquecimiento semantico **entonces**
5: $SS = encontrarSimilitudSemantica(T_q, T_t, DO)$ (Algoritmo 12)
6: $SS_I = econtrarSimilitudSemanticaEntradas(T_q, T_t)$ (Algoritmo 13)
7: $SS_O = econtrarSimilitudSemanticaSalidas(T_q, T_t)$ (Algoritmo 13)
8: $TSS = SS + SS_I + SS_O$
9: **devolver** $C_{tarea} = 1 - TSS$
10: **si no**
11: $LS = encontrarsimilitudLinguistica(T_q, T_t)$
12: **devolver** $C_{tarea} = 1 - LS$
13: **fin si**
14: **si no**, **si** $T_q \in TipoEvento$ **entonces**
15: $LS = encontrarsimilitudLinguistica(T_q, T_t)$
16: **devolver** $C_{tarea} = 1 - LS$
17: **fin si**

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.

$$St_{sim} = \frac{1}{1 + TD|} \tag{4.8}$$

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.

$$LN_{sim} = \frac{St_{sim} * nodosIntersectados}{nodosConsulta} \tag{4.9}$$

4.4.3. Similitud de comportamiento secuencial (SBsim)

Evalúa 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 ($nSeqT$).

$$SB_{sim} = \frac{St_{sim}}{1 + nSeqQ + nSeqT - 2 * nSeqT * nSeqQ} \tag{4.10}$$

Utilizando estas tres medidas de similitud, los BP del repositorio son clasificados en tres rankings desde el más similar hasta el menos similar respecto a un grafo de consulta. Los rankings son presentados al usuario final a través del módulo organizador de ranking estructural (Figura 4.3(e)), el cual tiene una interfaz gráfica de usuario que facilita la visualización de: los resultados de similitud, las operaciones de edición ejecutadas, y el grafo de resultado en comparación con el grafo de consulta.

4.5. Resumen

Este capítulo describió el módulo analizador estructural y semántico el cual ejecuta un análisis de correspondencia estructural entre un BP de consulta y un conjunto de BP del repositorio (cada BP del ranking del repositorio). El análisis tiene en cuenta una serie de operaciones de edición sobre cada uno de los BP del repositorio con el fin de hacerlos tan similares como sea posible a un BP de consulta. Como resultado, el analizador estructural y semántico entrega un ranking de grafos de resultado y su correspondencia con el BP de consulta.

Capítulo 5

Resultados y Discusión

Este capítulo presenta los materiales y métodos utilizados en la presente aproximación científica para evaluar experimentalmente el entorno de recuperación de BP basado en semántica del comportamiento. La evaluación experimental fue realizada por medio de dos análisis; el primero estudia el rendimiento del prototipo, y el segundo su relevancia. Los resultados de relevancia fueron utilizados para comparar las medidas de precisión y exhaustividad del prototipo *BeMantics* respecto a un trabajo previo denominado *BeMatch (A Platform for Matchmaking Service Behavior Models)* desarrollado por Corrales, Grigori y cols., (Corrales, Grigori, Bouzeghoub, y Burbano, 2008). Finalmente, este capítulo detalla los resultados de la evaluación.

5.1. Materiales y métodos

5.1.1. Conjunto de procesos de negocio para pruebas

Para probar el entorno de recuperación de BP basado en semántica de comportamiento fue desarrollado un conjunto de BP (conjunto de prueba) utilizando el lenguaje de modelado BPMO. Este conjunto está compuesto por 60 BP del dominio de telecomunicaciones y 40 BP del dominio del geo-procesamiento. En este caso solo los BP de telecomunicaciones fueron enriquecidos semánticamente utilizando las ontologías de dominio *seTOM* y *sSID*.

5.1.2. Modelo de evaluación de pertinencia

Para evaluar el entorno *BeMantics* en términos de relevancia de BP se diseñó un modelo de evaluación de pertinencia (Figueroa, Sandino, y Corrales, 2011) el cual permite que un conjunto de 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 (conjunto de BP relevantes).

Criterios de evaluación

Los criterios de evaluación proporcionan valores numéricos con los cuales los jueces humanos emiten juicios de semejanza entre los BP para obtener conjuntos jerarquizados de relevancia confiables. El modelo descrito en la presente investigación científica permitió clasificar los criterios para la evaluación intuitiva en cuatro niveles: estructura, comportamiento, semántica e interfaces, cada uno de estos niveles fue valorado utilizando juicios de semejanza para el emparejamiento de BP establecidos en trabajos previos (Wombacher y Li, 2010; Bernstein, Kaufmann, Bürki, y Klein, 2005; Goderis y cols., 2009).

- **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:* ofrece una idea visual de la secuencia de las tareas dentro de un BP. Para la evaluación de este criterio fue establecida la siguiente escala: *exacto* cuando gráficamente el BP de consulta y el BP del repositorio son iguales; *complemento* cuando el BP del repositorio está contenido, es parte o hereda estructuras del BP de consulta; *inclusión* cuando el BP de consulta está contenido, es parte o hereda del BP del repositorio; *inexacta* cuando el BP del repositorio presenta estructuras que no son exactamente iguales al BP de 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 (Hidders y cols., 2005). 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 (Paolucci y cols., 2002). 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 de consulta existe una y sólo una tarea de un BP del repositorio cuya lista de entradas posea elementos semejantes.

- *salidas:* este criterio evalúa las salidas de los BP de manera similar a la evaluación de las entradas.

Medidas de recuperación de los BP

Las medidas de recuperación evalúan los criterios en términos del rendimiento y la relevancia de recuperación (Blair, 1990; Baeza-Yates y Ribeiro-Neto, 1999; Borlund, 2000; Pors, 2000). El rendimiento está relacionado con el tiempo de respuesta; y la relevancia con la efectividad de recuperación, la cual se puede estimar utilizando unas medidas conocidas como precisión (P) y exhaustividad (R).

La precisión evalúa la habilidad de una herramienta de descubrimiento automática para recuperar solamente elementos relevantes (aquellos elementos considerados como similares a consultas realizadas por jueces humanos) y evitar resultados inesperados o falsos positivos (aquellos elementos recuperados no relevantes); y la exhaustividad evalúa la habilidad para recuperar todos los elementos relevantes evitando falsos positivos (elementos relevantes no recuperados). Estas medidas pueden clasificarse en binarias o gradadas.

Las medidas de efectividad binarias (P y R) permiten relacionar el número de elementos relevantes recuperados ($BPr \cap BPe$) con la cantidad total de elementos recuperados BPe y de elementos relevantes BPr (en este caso solo se consideran dos niveles de relevancia: relevante y no relevante). Las ecuaciones 5.2 Y 5.1 muestran las expresiones matemáticas para la evaluación binaria (Baeza-Yates y Ribeiro-Neto, 1999).

$$R = \frac{BPr \cap BPe}{BPr} \tag{5.1}$$

$$P = \frac{BPr \cap BPe}{BPe} \tag{5.2}$$

Por su parte, las medidas de efectividad gradadas (Pg y Rg) (Küster y König-Ries, 2008) proporcionan una clasificación de los BP del repositorio (T_i) considerados similares a un BP de consulta (Q) de acuerdo con diferentes niveles de relevancia. Es decir, mientras P y R solo consideran la cantidad de elementos relevantes y recuperados; Pg y Rg tienen en cuenta la suma total de grados de relevancia entre los BP. Las ecuaciones 5.3 y 5.4 muestran las expresiones matemáticas utilizadas para la evaluación gradada.

$$Rg = \frac{\sum_{T_i \in T} min\{f_r(Q, T_i), f_e(Q, T_i)\}}{\sum_{T_i \in T} f_r(Q, T_i)} \tag{5.3}$$

$$Pg = \frac{\sum_{T_i \in T} min\{f_r(Q, T_i), f_e(Q, T_i)\}}{\sum_{T_i \in T} f_e(Q, T_i)} \tag{5.4}$$

La ecuación 5.3 relaciona la sumatoria de todos los valores mínimos entre los resultados de una herramienta automática (f_e) y los resultados de la evaluación de los jueces (f_r), con la sumatoria de todos los resultados de la evaluación de los jueces. De manera

análoga la ecuación 5.4 relaciona los valores mínimos entre la herramienta automática y la evaluación de los jueces con la sumatoria de todos los resultados de la herramienta automática. De esta forma Pg y Rg tienen en cuenta la suma total de grados de relevancia entre los BP.

En la presente aproximación científica la evaluación de la relevancia del entorno *"BeMantics"* fue realizada a través del modelo gradado. En este caso BeMantics comparó cada elemento T_i del conjunto de cien BP del repositorio, con cada uno de siete BP de consulta Q predefinidos, con el fin de obtener un ranking automático f_e por cada BP de consulta. Luego, mediante el modelo para la evaluación de la pertinencia (Figueroa y cols., 2011), los jueces humanos emitieron evaluaciones de relevancia de los BP del repositorio obteniendo como resultado un ranking real f_r respecto a cada uno de siete BP de consulta.

5.2. Resultados

5.2.1. 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 del repositorio; y los tiempos de almacenamiento y generación de la lista ordenada de BP del repositorio. Para este análisis fue empleado un servidor el cual alojó la implementación del entorno BeMantics desarrollada en el lenguaje de programación Java. Las características hardware y software de dicho servidor están resumidas en la tabla 5.1.

Característica	**Valor**
RAM	4GB
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

Repositorio de BP basado en semántica del comportamiento

Para analizar la ejecución del repositorio fue almacenado un conjunto de 100 BP y para probar el sistema de clasificación fueron seleccionados 6 procesos de consulta. La figura 5.1 muestra la evaluación del almacenamiento de BP en el repositorio de acuerdo al número de nodos. En ella puede observarse que el tiempo consumido para almacenar un BP en el repositorio varía entre los 15 y 25 segundos manteniendo una tendencia lineal de crecimiento en dependencia del número de nodos de cada BP.

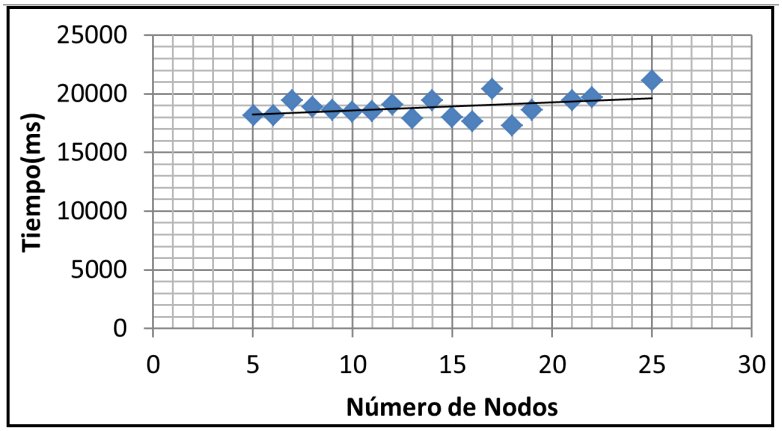


Figura 5.1: Evaluación del rendimiento del almacenamiento de un BP en el repositorio de acuerdo a su número de nodos. Fuente propia.

Por otra parte, la figura 5.2 presenta la evaluación del rendimiento para la fase de recuperación de BP del repositorio de acuerdo a la semántica del comportamiento.

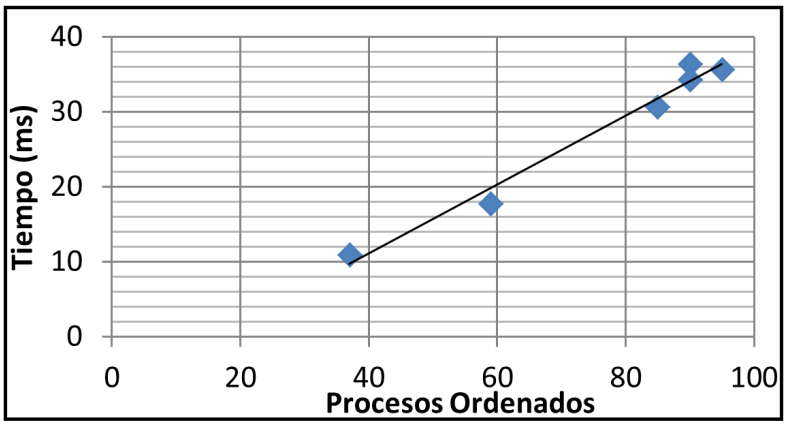


Figura 5.2: Evaluación del rendimiento de recuperación de BP ejecutada en el repositorio. Fuente propia.

Los resultados de ésta figura, muestran que el tiempo consumido para generar una lista ordenada de 40 a 90 BP varía entre 10 y 40ms. Los valores tan cortos de tiempo son debidos a que la fase de recuperación de BP del repositorio está basada en patrones de flujo de control los cuales están referenciados por etiquetas incluidas en los BP de una base de datos relacional; en la cual la consulta de patrones está simplificada a una simple consulta a la base de datos. Esto es así, dado que la detección de patrones de flujo de control para los BP del repositorio es realizada en la fase de almacenamiento y en la fase de recuperación solamente es ejecutada para el BP de consulta.

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
Susstituir 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 fue seleccionado experimentalmente. En este caso se probaron diferentes valores de AC con el fin de determinar un valor apropiado para encontrar un alto número de correspondencias posibles en un tiempo aceptable.

La figura 5.3 presenta la evaluación de rendimiento del analizador estructural y semántico con diferentes valores de AC; y la figura 5.4 muestra los valores de AC y el número de correspondencias obtenidas con los 6 procesos de consulta $(Q_1, ..., Q_6)$. A partir de estos resultados puede concluirse que 5 es el mejor valor para AC debido a que permitió encontrar un consumo intermedio de tiempo y un alto número de correspondencias.

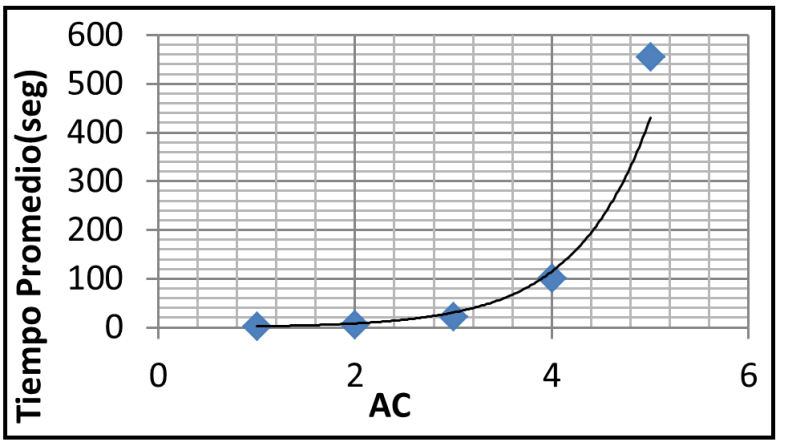


Figura 5.3: Selección del parámetro costo aceptable (AC) de acuerdo con el consumo de tiempo. Fuente propia.

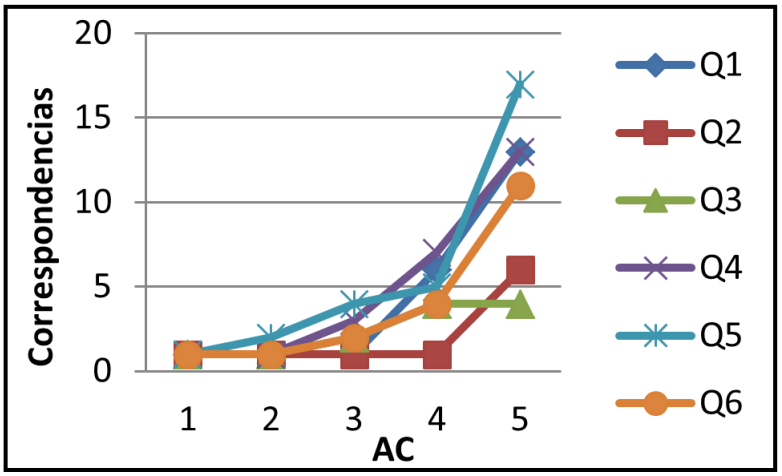


Figura 5.4: Selección del parámetro costo aceptable (AC) de acuerdo con el número de correspondencias. Fuente propia.

Por otra parte, la figura 5.3 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 5.4 muestra que para el valor 5 fueron obtenidos valores altos de correspondencias para todas las consultas. No obstante, para valores inferiores fue observada una reducción notable en este valor y para valores superiores a 5, aunque no aparece en la gráfica, fue encontrado que el consumo de tiempo creció de manera exagerada. Por lo tanto, en el entorno BeMantics, el algoritmo de recuperación estructural y semántica fue ejecutado con el valor 5 para AC obteniendo la figura 5.5 la cual exhibe el promedio de consumo de tiempo en contraste con el promedio de nodos de los BP del repositorio.

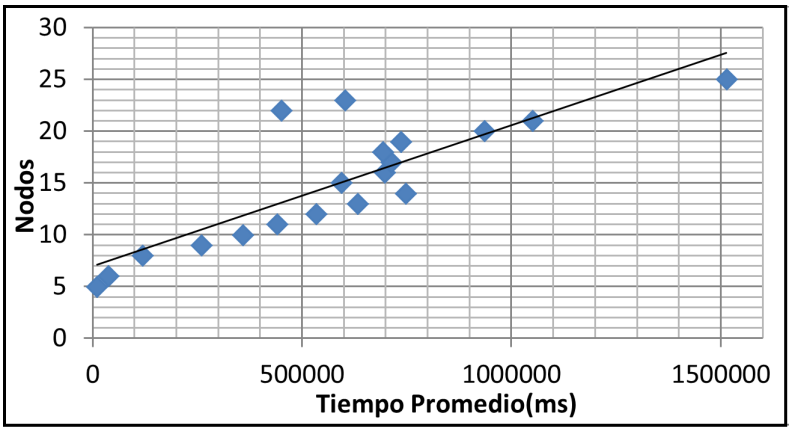


Figura 5.5: Consumo de tiempo en la correspondencia vs el número de nodos de los BP. Fuente propia.

La figura 5.5 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 del 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.

5.2.2. Análisis de relevancia

Este análisis fue ejecutado con el propósito de determinar las medidas gradadas promedio de Pg y Rg 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 flujo de control; el segundo llamado BeMatch, 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 adicionando un repositorio que permite ejecutar un mecanismo de indexación (pre-correspondencia).

De esta manera, en el análisis de relevancia, fueron comparados los valores de Pg y Rg entre los tres prototipos de acuerdo a criterios de estructura, lingüística y comportamiento secuencial, tal como lo muestran las figuras 5.6 y 5.7.

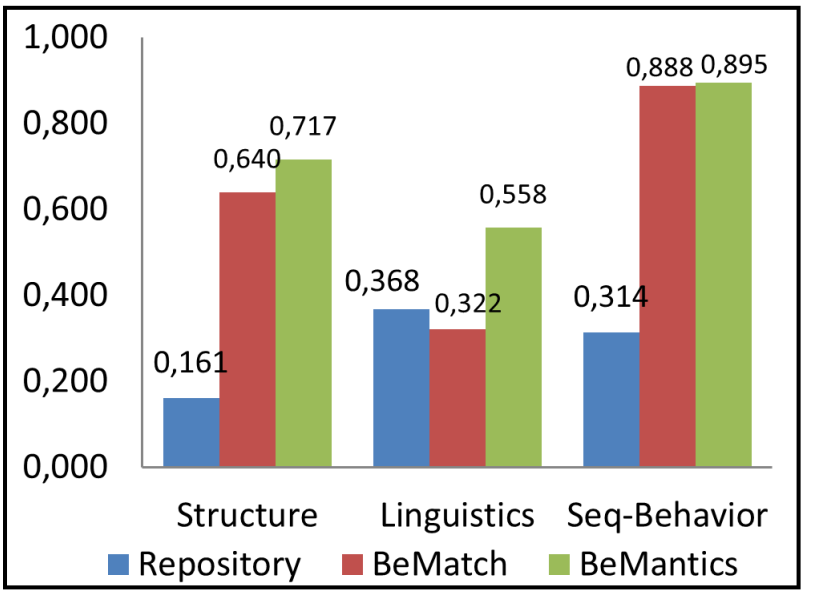


Figura 5.6: Valores promedio de precisión gradada para el repositorio, BeMantics y BeMatch.
Fuente propia.

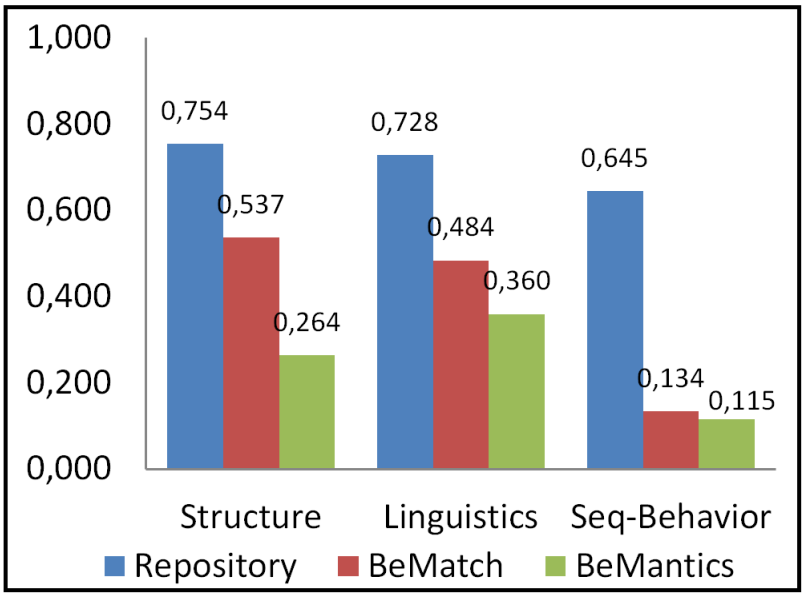


Figura 5.7: Valores promedio de exhaustividad gradada para el repositorio, BeMantics y BeMatch. Fuente propia.

Comparando la precisión gradada (figura 5.6), 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 (valor mínimo de fasos positivos). Además, puede observarse que el criterio que mejor fue valorado en BeMantics respecto a BeMatch es 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 recuperados no fueron relevantes).

Sin embargo, comparando los valores para la exhaustividad gradada (figura 5.7), 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 fueron recuperados) debido a que algunos procesos relevantes fueron perdidos en la fase de pre-correspondencia ejecutada por el repositorio, aun cuando este mostró valores altos de Rg.

Finalmente, en los resultados del repositorio puede observarse que el repositorio alcanzó el mejor valor de exhaustividad Rg y el peor valor de precisión Pg. Esto es normal dado que el repositorio está diseñado como un filtro inicial con un alto valor de Rg (valores bajos de falsos negativos); y en este caso específico el valor de Pg no es tan importante, debido a que se supone que los algoritmos de correspondencia deben mejorar la precisión, tal como puede observarse en la figura 5.7.

5.3. Resumen

En este capítulo fueron presentados los métodos y materiales utilizados para evaluar el rendimiento y relevancia del entorno BeMantics en comparación con otra herramienta denominada BeMatch. Además, se presentaron los resultados de dicha evaluación, los cuales demostraron que el protitipo más preciso fue BeMantics aún cuando obtuvo el valor más bajo de exhaustividad. En cuanto al repositorio de BP basado en semántica del comportamiento, obtuvo valores bajos de prcisión, pero con valores altos de exhaustividad, lo cual permitió concluir que el repositorio recuperó BP no relevantes, pero redujo el número de falsos negativos.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

La búsqueda de BP es un campo crítico en muchas áreas, debido a la dificultad de encontrar componentes que puedan adaptarse exactamente a las exigencias de los usuarios. Por esta razón, el desarrollo de nuevos métodos de búsqueda intuitivos, dotados de inteligencia artificial, basados en semántica y que reconozcan lo que realmente el usuario necesita recuperar es un área importante de la I+D para agilizar el despliegue y configuración de nuevos BP.

El presente libro presenta los resultados de una investigación científica, llevada a cabo por el Grupo de Ingeniería Telemática de la Universidad del Cauca (Colombia), en la cual fue generado un entorno denominado BeMantics para la recuperación de BP basado en características estructurales, lingüísticas y de semántica del comportamiento. La semántica del comportamiento, en este contexto, corresponde a un método de indexación fundamentado en los patrones de flujo de control y sus relaciones semánticas. De esta manera, el diseño de BeMantics utilizó dos módulos principales; primero un repositorio con mecanismos de pre-correspondencia basado 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 BP de consulta con cada uno de los BP obtenidos del repositorio y ejecuta operaciones de edición sobre ellos con el fin de hacerlos tan similares como sea posible al BP de consulta.

Para analizar los resultados del entorno BeMantics fue desarrollada 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. Esta evaluación permitió comparar los resultados de BeMantics con un trabajo previo denominado BeMatch, el cual contiene un analizador estructural similar, pero carece del módulo de pre-correspondencia y del módulo de análisis

semántico sobre la tareas de los BP.

La evaluación de relevancia para el entorno Bemantics demostró que la aplicación de la semántica en la comparación de las tareas de los BP permitió incrementar la precisión gradada pero redujo la exhaustividad respecto a BeMatch el cual solo ejecutó una comparación léxica. Por lo tanto, puede afirmarse 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 puede obtenerse al utilizar ontologías de dominios más generales, ya que el conjunto de conceptos con los que son etiquetados los nombres e interfaces de las tareas está reducido al conjunto de conceptos incluidos en la ontología de dominio.

Por consiguiente, la presente investigación recomienda el uso de ontologías de dominios más generales debido a que ofrecen un grupo más abierto de concepto, no obstante si el contexto dónde se ejecutan los BP está muy bien definido una ontología de dominio específico puede reducir ambigüedades entre los profesionales que se encarguen de la fase de enriquecimiento semántico de los BP.

Por otra parte, la evaluación de rendimiento demostró que el repositorio presentó un bajo consumo de tiempo (10-40ms) y consecuentemente al utilizar el mecanismo de ranking basado en semántica del comportamiento como fase de pre-correspondencia es posible reducir el espacio de búsqueda y en general el consumo total de tiempo para el entorno BeMantics.

6.2. Trabajo futuro

A continuación están listados los trabajos futuros planteados como continuación de la aproximación científica descrita en este libro.

- **Incrementar los métodos de recuperación de BP:** como trabajo futuro pueden aplicarse mecanismos de minería de BP, de tal manera que el sistema no solo descubra BP en los cuatro niveles previamente mencionados, sino que además, "infiera" las preferencias de los usuarios y las patrones de comportamiento con mayor ocurrencias de ejecución dentro de los BP; permitiendo de esta manera realizar una búsqueda personalizada de servicios.

- **Implementación de una ontología que contenga las relaciones existentes entre todos los patrones de flujo de control de referencia viables en BPMO:** la ontología implementada (bpmooner-1.0.0.wsml) contempla solamente 12 patrones básicos de BPMO, sin embargo, puede extenderse para que soporte la totalidad de los 21 patrones de flujo de control de referencia viables en BPMO.

- **Adicionar soporte para más lenguajes de modelado a la capa de transformación de BP:** a partir de la arquitectura propuesta es posible implementar mecanismos de trans-

formación que soporten diferentes notaciones de modelado de BP, tales como BPMN, XPDL, BPEL, entre otros.

- **Plataforma para búsqueda y composición de BP semánticos:** es posible crear una plataforma para facilitar la gestión (crear, editar, eliminar, buscar y modificar) de los modelos de BP, la cual incluya un entorno de composición de BP utilizando fragmentos de otros BP descubiertos en un repositorio de BP.
- **Experimentación del entorno BeMantics en un entorno empresarial real:** como trabajo futuro se propone 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:** para mejorar el rendimiento del mecanismo de correspondencia estructural, es posible utilizar técnicas basadas en heurísticas que reduzcan el tiempo de ejecución del analizador estructural y semántico.
- **Mejorar la plataforma de evaluación de pertinencia de la recuperación de BP:** en este caso puede desarrollarse un módulo de evaluación de efectividad que reciba directamente los resultados de las herramientas automáticas; los compare con los valores de relevancia obtenidos de los juicios de semejanza emitidos por jueces humanos; aplique las ecuaciones para medir la efectividad de la recuperación; y construya gráficas que describan esta medición. Este nuevo módulo ahorraría tiempo en análisis de datos y facilitaría la valoración de la efectividad de un algoritmo.

6.3. Resumen

Este último capítulo expuso las principales conclusiones obtenidas de la investigación científica descrita en este libro y propuso una lista de trabajos futuros. Entre las conclusiones obtenidas puede destacarse que la creación de un filtro de semántica del comportamiento permite obtener un mejoramiento en cuanto al tiempo de ejecución del sistema en general, conservando, en cierto modo, la relevancia de los resultados de correspondencia. Este último capítulo expuso las principales conclusiones obtenidas de la investigación científica descrita en este libro y propuso una lista de trabajos futuros. Entre las conclusiones obtenidas puede destacarse que la creación de un filtro de semántica del comportamiento permite obtener un mejoramiento en cuanto al tiempo de ejecución del sistema en general, conservando, en cierto modo, la relevancia de los resultados de correspondencia.

Referencias

Baeza-Yates, R. A., y Ribeiro-Neto, B. (1999). *Modern information retrieval*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Bechhofer, S., Goble, C., y Horrocks, I. (2001). *Daml+oil is not enough.*

Belecheanu, R., Cabral, L., Domingue, J., Gaaloul, W., Hepp, M., Filipowska, A., y cols. (2007). *Business process ontology framework* (Inf. Téc. n.º Deliverable 1.1). SUPER.

Benatallah, B., Hacid, M., Rey, C., y Toumani, F. (2003). Semantic reasoning for web services discovery.

Benavides, A., Enriquez, G., Ramirez, J. D., Figueroa, C., y Corrales, J. C. (2012). *Control-flow patterns in converged services.*

Bernstein, A., Kaufmann, E., Bürki, C., y Klein, M. (2005). How similar is it? towards personalized similarity measures in ontologies. En *Internationale tagung wirtschaft-sinformatik* (pp. 1347–1366).

Bhakti, M. A. C., y Abdullah, A. B. (2011). Formal modelling of an autonomic service oriented architecture.

Black, P. E. (2008). *Finite state machine.* Disponible en `http://xlinux.nist.gov/dads/HTML/finiteStateMachine.html`

Blair, D. C. (1990). *Language and representation in information retrieval*. New York, NY, USA: Elsevier North-Holland, Inc.

Borlund, P. (2000). Experimental components for the evaluation of interactive information retrieval systems. *Journal of Documentation*, *56*, 71–90.

Brickley, D., y Guha, R. (2002). *RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft* (Inf. Téc.). http://www.w3.org./TR/rdf-schema/: W3C.

Bruijin, J. d., y Heymans, S. (2006, Diciembre 18, 2006). *D1.7 wsml semantics specification* (Inf. Téc.). DIP.

Burrows, M., y Pendleburry, M. (2008). *etom, itil & iso/iec20000 harmonization.* Disponible en `http://www.bsmimpact.com/pdf/etom-itil-iso20k-harmonisation.pdf`

Castells, P. (2003). La web semántica. *Sistemas Interactivos y Colaborativos en la Web*, 195-212.

Choi, I., Kim, K., y Jang, M. (2007). An xml-based process repository and process query language for integrated process management. *Knowledge and Process Management*, *14*(4), 303–316.

Châtel, P. (2007). Toward a semantic web service discovery and dynamic orchestation ba-

sed on the formals specification of functional domain knowledge. *20th International Conference on Software & Systems Engineering and their Applications*.

Chu, K., Cordero, O., Korf, M., Pickersgill, C., y Whitmore, R. (2006). Introduction to soa and the oracle soa suite. En Oracle (Ed.), *Oracle soa suite developer's guide 10g (10.1.3.1.0)* (p. 1.1 - 1.10). Redwood City, CA: Oracle.

Corcho, O., y Gómez-Pérez, A. (2000). A roadmap to ontology specification languages. En *12th european workshop on knowledge acquisition, modeling and management* (pp. 80–96). London, UK, UK: Springer-Verlag.

Corrales, J. C. (2008). *Behavioral matchmaking for service retrieval.* Computer science, University of Versailles Saint-Quentin-en-Yvelines.

Corrales, J. C., Grigori, D., y Bouzeghoub, M. (2006). Bpel processes matchmaking for service discovery. *On the Move to Meaningful Internet Systems 2006 CoopIS DOA GADA and ODBASE*, *4275*, 237–254.

Corrales, J. C., Grigori, D., Bouzeghoub, M., y Burbano, J. E. (2008). Bematch: a platform for matchmaking service behavior models. En *11th international conference on extending database technology: Advances in database technology* (p. 695-699). New York, NY, USA: ACM.

Corrales, J. C., Grigori, D., Bouzeghoub, M., y Gater, A. (2010). Ranking bpel processes for service discovery. *IEEE Transactions on Services Computing*, *3*, 178-192.

CRM-Management-Editors. (2009, 09/10/2009). *Philippine telco cuts time to market for new services by 80 percent* (Vol. 2011) (n.º 06/12/2011). Enterprise Innovation. Disponible en `http://www.enterpriseinnovation.net/content/philippine-telco-cuts-time-market-new-services-80`

Daciuk, J., Watson, B. W., Mihov, S., y Watson, R. E. (2000). Incremental construction of minimal acyclic finite-state automata. *Comput. Linguist.*, *26*, 3–16.

Dawidziuk, R., y Cieśĺak, P. (2007, Diciembre). *A proof-of-concept ontology of telecommunication services* (Deriverable n.º 3.5). SIMS Consortium. Disponible en `http://meag.tele.pw.edu.pl/sims/deliverables/SIMS%20Deliverable%203_5.pdf`

de Medeiros, A. K. A., Dongen, B. F. van, y van der Aalst, W. M. P. (2004). Process mining : Extending the alpha-algorithm to mine short loops. *Technology*. Disponible en `http://alexandria.tue.nl/repository/books/576199.pdf`

de Medeiros, A. K. A., Weijters, A. J. M. M., y van der Aalst, W. M. P. (2007, Abril). Genetic process mining: an experimental evaluation. *Data Min. Knowl. Discov.*, *14*, 245–304.

Decker, G., Overdick, H., y Weske, M. (2008). Oryx -an open modeling platform for the bpm community. En *6th international conference on business process management* (pp. 382–385). Berlin, Heidelberg: Springer-Verlag.

Devitt, A., Danev, B., y Matusikova, K. (2006). Ontology-driven automatic construction of bayesian networks for telecommunication network management.

Dimitrov, M., Momtchev, V., Simov, A., Ognyanoff, D., y Konstantinov, M. (2006, Nov. 16th 2006). *wsmo4j programmers guide v. 2.0.1* (Vol. 2011) (n.º Nov. 24th). Ontotext.

Dubray, J.-J. (2008). *Bpml* (Vol. 2011) (n.º 20/11/2011). Disponible en `http://www.ebpml.org/bpml.htm`

Elhadad, M., Balaban, M., y Sturm, A. (2008). Effective business process outsourcing: The

prosero approach. *International Journal of Interoperability in Business Information Systems*, *3*(1).

Eshuis, R., y Grefen, P. W. P. J. (2007). Structural matching of bpel processes. En (p. 171-180). IEEE Computer Society.

Estero, F., A.; Palomo, y Medina Bulo, I. (2008). Operadores de mutación para ws-bpel 2.0. En *Actas del iii taller sobre pruebas en ingeniería del software* (Vol. 2, pp. 13–18).

Fensel, D., Horrocks, I., Harmelen, F. van, Decker, S., Erdmann, M., y Klein, M. (2000). Oil in a nutshell. En (p. 1-16). Springer.

Ferreiro, D. (2002). Integración cooperativa. *Jornada Temática - Empresa Agraria y Cooperativismo"Madrid, 2002*.

Ferro, A., Giugno, R., Pulvirenti, A., Recupero, D. R., Shasha, D., y Skripin, D. (2007). Graphblast: Multi-feature graphs database searching. *Analysis*, *XX*, 1–6.

Fielding, R. T., y Taylor, R. N. (2000). Principled design of the modern web architecture. En (p. 407-416). ACM.

Figueroa, C., Sandino, L., y Corrales, J. C. (2011, 05/2011). Plataforma para evaluar sistemas de recuperación de procesos de negocio. *Revista de Investigaciones UCM*, *17*, 64-76.

Fleck, J. (2003). *Overview of the structure of the ngoss architecture* (Inf. Téc.). Hewlett-Packard Company.

Floch, J. (2007). Techniques for ontology-driven semantic interface artefacts, final version. *Techniques*(1).

Frankowski, J., Rubach, P., y Szczekocka, E. (2007). Collaborative ontology development in real telecom environment. En W. Abramowicz y L. A. Maciaszek (Eds.), (Vol. 116, p. 40-53).

Fronk, M., y Lemcke, J. (2006). Expressing semantic web service behavior using description logics. *European Semantic Web Conference (ESWC). 2006.*.

García Sánchez, F. (2008). *Sistema basado en tecnologías del conocimiento para entornos de servicios web semánticos*. Tesis Doctoral no publicada, Universidad de Murci.

Gater, A., Grigori, D., y Bouzeghoub, M. (2010). Owl-s process model matchmaking. En *Ieee international conference on web services* (p. 640-641). IEEE Computer Society.

Ge, J., y Qiu, Y. (2008). Concept similarity matching based on semantic distance. En *4th international conference on semantics, knowledge and grid* (pp. 380–383). Washington, DC, USA: IEEE Computer Society.

Giugno, R., y Shasha, D. (2002). Graphgrep: A fast and universal method for querying graphs. En *Pattern recognition, 2002. proceedings. 16th international conference on* (Vol. 2, p. 112-115).

Goderis, A., Fisher, P., Gibson, A., Tanoh, F., Wolstencroft, K., De Roure, D., y cols. (2009, Noviembre). Benchmarking workflow discovery: a case study from bioinformatics. *Concurr. Comput. : Pract. Exper.*, *21*, 2052-2069.

Goedertier, S., De Weerdt, J., Martens, D., Vanthienen, J., y Baesens, B. (2011). Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, *11*(2), 1697-1710.

Goedertier, S., Martens, D., Vanthienen, J., y Baesens, B. (2009, Junio). Robust process discovery with artificial negative events. *J. Mach. Learn. Res.*, *10*, 1305–1340.

Gonçalves da Silva, E., Ferreira Pires, L., y van Sinderen, M. (2011). Towards runtime

discovery, selection and composition of semantic services. *Computer Communications*, 159-168.

Grimm, S., Keller, U., Lausen, H., y Nagypál, G. (2007). A reasoning framework for rule-based wsml. En *4th european conference on the semantic web: Research and applications* (p. 114-128). Berlin, Heidelberg: Springer-Verlag.

Guerra M, C. (2002). *Fusiones bancarias - el caso bancomer*. Tesis Doctoral no publicada, Universidad Autónoma de México.

Havey, M. (2005). The pi-calculus. En *Essential business process modeling* (p. 122). O'Reilly Media.

Hendler, J., y McGuiness, D. (2000). The darpa agent markup language. *IEEE Intelligent Systems*(Trends and Controversies), 6-7.

Hendrickson, B., y Leland, R. (1995). A multilevel algorithm for partitioning graphs. En *Acm/ieee conference on supercomputing* (p. 28). ACM.

Hewlet-Packard. (2009). *Integración empresarial.* Hewlett-Packard Development Company, L.P. Disponible en `http://www.hp.com/latam/ar/servicios/aplicaciones_empresariales/enter_empresarial.html`

Hidders, J., Dumas, M., van der Aalst, W. M. P., Hofstede, A. H. M. ter, y Verelst, J. (2005). When are two workflows the same? En (Vol. 41, p. 3-11). Australian Computer Society.

Hollingsworth, D. (1995). *The workflow reference model version 1.1.* The WfMC.

IBM. (2008). *Service oriented architecture - soa.* Autor. Disponible en `http://www-01.ibm.com/software/solutions/soa/`

Inter-American Development Bank, y Oranisation for Economic Co-operation and Development. (2005). *Concentraciones empresariales 2000-225 analizadas por la fiscalía nacional económica de chile* (n.º Foro Latinoamericano de Compentencia). Foro Latinoamericano de Compentencia.

ITU. (2005). *Enhanced telecom operations map (etom).*

Jin, T., Wang, J., Wu, N., La Rosa, M., y Ter Hofstede, A. H. M. (2010). Efficient and accurate retrieval of business process models through indexing. En *International conference on on the move to meaningful internet systems - volume part i* (pp. 402–409). Berlin, Heidelberg: Springer-Verlag.

Jordan, D., y Alves, A. (2007). Web services business process execution language version 2 . 0. *Language*, *11*(Abril), 1–264.

Kaczmarek, M., Konstantinov, M., Ma, Z., Wieloch, K., y Zebrowski, P. (2008). *Deliverable 3.4. business process library final prototype* (Inf. Téc.). SUPER.

Keller, G., Nuttgens, M., y Scheer, A. (1992). *Semantische processmodellierung auf der grundlage ereignisgesteuerter processketten (epk).* Univ. of Saarland.

Kiefer, C., Bernstein, A., Lee, H. J., Klein, M., y Stocker, M. (2007). Semantic process retrieval with isparql. En *4th european conference on the semantic web: Research and applications* (pp. 609–623). Berlin, Heidelberg: Springer-Verlag.

Klusch, M., Fries, B., y Sycara, K. (2006). Automated semantic web service discovery with owls-mx. En *5th international joint conference on autonomous agents and multiagent systems* (pp. 915–922). New York, NY, USA: ACM.

Ko, R. K. L., Lee, S. S. G., y Lee, E. W. (2009). Business process management (bpm)

standards: a survey. *Business Process Management Journal*, *15*(5), 744-791.

Kokash, N., Heuvel, W. van den, y D'Andrea, V. (2006). Leveraging web services discovery with customizable hybrid matching. En (p. 522-528).

Kopecky, J., Vitvar, T., Bournez, C., y Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, *11*, 60-67.

Koschmider, A., Hornung, T., y Oberweis, A. (2011, Junio). Recommendation-based editor for business process modeling. *Data Knowl. Eng.*, *70*, 483-503.

Küster, U., y König-Ries, B. (2008). On the empirical evaluation of semantic web service approaches: Towards common sws test collections. En *2nd ieee international conference on semantic computing* (p. 339-346). IEEE Computer Society.

Lambea, J. (2009, Octubre). *Business sla management* (Deriverable n.º D.A2a). SLA@SOI. Disponible en `http://sla-at-soi.eu/wp-content/uploads/2009/11/D.A2a-M12-Business_SLA_Managament-M17.pdf`

La Rosa, M., Reijers, H. A., van der Aalst, W. M. P., Dijkman, R. M., Mendling, J., Dumas, M., y cols. (2011, Junio). Apromore: An advanced process model repository. *Expert Syst. Appl.*, *38*, 7029–7040.

Lausen, H., Bruijn, J. de, Polleres, A., y Fensel, D. (2005). Wsml - a language framework for semantic web services. En *Rule languages for interoperability.* W3C.

Lin, L., y Arpinar, I. B. (2006). Discovery of semantic relations between web services. *IEEE International Conference on Web Services*, 357-364.

Livanos-Cattaui, M. (2003, Diciembre). *El papel "vital" de las empresas.* BBC. Disponible en `http://news.bbc.co.uk/hi/spanish/specials/2003/la_brecha_digital/newsid_3290000/3290781.stm`

Lu, R., Sadiq, S., y Governatori, G. (2009). On managing business processes variants. *Data Knowledge Engineering*, *68*, 642–664.

Mahleko, B., y Wombacher, A. (2006). Indexing business processes based on annotated finite state automata. En *The ieee international conference on web services* (p. 303-311). IEEE Computer Society.

Manola, F., y Miller, E. (Eds.). (2004, Februar). Rdf primer. Disponible en `http://www.w3.org/TR/2004/REC-rdf-primer-20040210/` (Stand: 15.4.2009)

Markovic, I. (2009, 02.09.2009 Bulletin 2009/36). *Enhanced process query framework.*

Markovic, I., Pereira, A. C., y Stojanovic, N. (2008). A framework for querying in business process modelling. En M. Bichler y cols. (Eds.), *Multikonferenz wirtschaftsinformatik.* GITO-Verlag, Berlin.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., y cols. (2004, Noviembre). *Owl-s: Semantic markup for web services.* Disponible en `http://www.w3.org/Submission/2004/07/`

Martínez, J., y Pérez, N. (2008, Octubre 29-31). *Yatosp: Marco de referencia semántico para el sector telco.*

McGuinness, D. L., y Harmelen, F. van. (2004). *Owl web ontology language overview* (Inf. Téc.). W3C.

Messmer, B. (1995). *Graph matching algorithms and applications*. Tesis Doctoral no publicada, University of Bern.

Miller, G. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38*(11), 39-41.

Mincomunicaciones-Colombia. (2008). *Plan nacional de tecnologías de la información y las comunicaciones 2008 - 2019.* Ministerio-de-Comunicaciones-de-Colombia.

Mongiello, M., y Castelluccia, D. (2006). Modelling and verification of bpel business processes. *Model-Based Methodologies for Pervasive and Embedded Software, International Workshop on*, *0*, 144-148.

Nayak, R., y Lee, B. (2007). Web service discovery with additional semantics and clustering. En *Ieee/wic/acm international conference on web intelligence* (pp. 555–558). Washington, DC, USA: IEEE Computer Society.

OASIS. (2011, Julio). *Reference architecture foundation for service oriented architecture version 1.0* (Vol. 2011; Inf. Téc.). Autor. Disponible en `http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra.pdf`

Okkyung, C., y Sangyong, H. (2008). Flexible rule-based web services system for users. *Next Generation Web Services Practices, International Conference on*, *0*, 1-4.

Oracle. (2010). *Oracle berkeley db java edition* (Vol. 2011) (n.º Nov. 16th). Autor. Disponible en `http://www.oracle.com/technetwork/database/berkeleydb/documentation/index-160410.html`

Ordóñez, L., Bastidas, A., y Corrales, J. C. (2012). Estimación de la similitud semántica de tareas entre dos procesos de negocio de telecomunicaciones. *Revista Ingeniería y Ciencia, Universidad EAFIT*, *8*(15), 65 - 95.

Ossenbruggen, J. R. van, Hardman, H. L., y Rutledge, L. (2002). Hypermedia and the semantic web: A research agenda. *JOURNAL OF DIGITAL INFORMATION*, *3*.

Paolucci, M., Kawamura, T., Payne, T. R., y Sycara, K. (2002). Semantic matching of web services capabilities.

Park, S.-H., Kim, S.-H., You, Y.-D., y Choi, H. (2007). Design and implementation of the high-speed encryption processing system for the variable sized data in cryptoki. En *International conference on convergence information technology* (pp. 1014–1019). Washington, DC, USA: IEEE Computer Society.

Patil, A. A., Oundhakar, S. A., Sheth, A. P., y Verma, K. (2004). Meteor-s web service annotation framework. En *13th international conference on world wide web* (pp. 553–562). New York, NY, USA: ACM.

P. Cordella, L., Foggia, P., Sansone, C., y Vento, M. (2004, Octubre). A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, *26*, 1367–1372.

Pedrinaci, C., Domingue, J., y Sheth, A. (2011). Semantic web services. En J. Domingue, D. Fensel, y J. Hendler (Eds.), *Handbook of semantic web technologies* (p. 977-1035). Springer Berlin Heidelberg.

Pollet, T., Maas, G., Marien, J., y Wambecq, A. (2006). *Telecom services delivery in a soa.* IEEE Computer Society.

Pors, N. O. (2000). Information retrieval, experimental models and statistical analysis. *Journal of Documentation*, *56*(1), 55-70. Disponible en `http://www.emeraldinsight.com/10.1108/EUM0000000007109`

Pérez C, N., Muñoz, H., Marcos, D. d. F., y Martínez E, J. (2008, 06/12/2011). Gestión

de procesos de negocio semánticos. Disponible en http://www.ip-super.org/res/Papers/SBPM-TelecomID2007.pdf

Qiao, X., y Li, X. (2010). Semantic web application for telecommunication services.

Ramirez, D., y Rojas, J. (2010). *Lineamientos para composición de servicios de telecomunicaciones en un entorno jain slee basados en software de libre distribución* [Tesis].

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

Rodriguez, R., y García, L. (2008). La gestión de los procesos de negocio en las empresas de telecomunicaciones.

Roman, D., Keller, U., Lausen, H., Bruijn, J. de, Lara, R., Stollberg, M., y cols. (2005). Web service modeling ontology. *Appl. Ontol.*, *1*, 77–106.

Rosen, K. (2006). Isomorphism of graphs. En M.-H. Science (Ed.), *Discrete mathematics and its applications* (6.ª ed., Vol. 6, p. 560-563). McGraw-Hill Science Engineering Math 6 edition.

Russell, N., H.M, A., Hofstede, t., van der Aalst, W. M. P., y Mulyar, N. (2006). *Workflow control-flow patterns: A revised view.* (Technical Report n.º BPM-06-22). BPM Center.

Sakr, S., y Awad, A. (2010). A framework for querying graph-based business process models. *19th international conference on World wide web WWW 10*, 1297-1297.

Sapkota, B. (2005). Web service discovery in distributed and heterogeneous environment. *WWW Service Composition with Semantic Web Services (wscomps05) Workshop, Compiegne, France*.

Sellami, M., Tata, S., y Defude, B. (2008). Service discovery in ubiquitous environments: Approaches and requirements for context-awareness. En (Vol. 17, p. 516-522). Springer.

Shangguan, Z., Gao, Z., y Zhu, K. (2008). Ontology-based process modeling using etom and itil. En L. Xu, A. Tjoa, y S. Chaudhry (Eds.), *Research and practical issues of enterprise information systems ii* (Vol. 255, p. 1001-1010). Springer Boston.

Shen, Z., y Su, J. (2005). Web service discovery based on behavior signatures. En *Ieee converence on service computing* (p. 279-286). IEEE Computer Society.

Silva-Muñoz, L. (2002). Representación de ontología en la web semántica. Disponible en http://www.geocities.ws/edincor2/articulo1.pdf

Song, M., Miller, J., y Arpinar, I. (2001). *Repox: Xml repository for workflow designs and specifications* (Inf. Téc.). University of Georgia.

Srinivasa, S., y Singh, M. H. (2005). Grace : A graph database system. En *International conference on management of data comad 2005b* (p. 4). Computer Society of India.

Stroulia, E., y Wang, Y. (2005). Structural and semantic matching for assessing web-service similarity. *Int. J. Cooperative Inf. Syst.*.

The-Open-Group. (2011). *The soa work group : Definition of soa* (Vol. 2011) (n.º 18-Nov). Autor. Disponible en http://www.opengroup.org/soa/soa/def.htm

The-WfMC. (1999). *Workflow management coalition terminology & glossary* (Inf. Téc.).

The-WfMC-Coalition. Disponible en http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

Tibco, S. I. U. (2006). *Extending the benefits of soa beyond the enterprise* (Inf. Téc.). Tibco Software Inc. UK.

TM-Forum. (2002). *Ngoss architecture technology neutral specification* (Vol. 2.5). TeleManagement Forum.

TM-Forum. (2005). *Information framework (sid) in depth* (Vol. 2010). Telemanagement Forum. Disponible en http://www.tmforum.org/InDepth/6647/home.html

Ullmann, J. R. (1976, Enero). An algorithm for subgraph isomorphism. *J. ACM*, *23*, 31–42.

van der Aalst, W. M. P. (1999). Formalization and verification of event-driven process chains. *Information and Software Technology*, *41*(10), 639-650.

van der Aalst, W. M. P., Hofstede, A. H. M., y Weske, M. (2003). Business process management: A survey. En *1st international conference on business process management* (Vol. 2678, p. 1-12). Springer-Verlag.

van der Aalst, W. M. P., Hofstede, A. H. M. T., Kiepuszewski, B., y Barros, A. P. (2003). Workflow patterns. *Distrib. Parallel Databases*, *14*(1), 5-51.

van der Aalst, W. M. P., Weijters, A. J. M. M., y Maruster, L. (2004). Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128-1142.

Vanhatalo, J. (2004). *Building and querying a repository of bpel process specifications* (Inf. Téc.). IBM Research GmbH.

Verma, K., Akkiraju, R., Goodwin, R., Doshi, P., y Lee, J. (2004). On accommodating inter service dependencies in web process flow composition. En *Aaai spring symposium on semantic web services* (pp. 37–43).

W3C. (2001a, Diciembre). *Guía breve de servicios web.* Disponible en http://w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

W3C. (2001b). *Web services description language (wsdl) 1.1.* Autor. Disponible en http://www.w3.org/TR/wsdl

W3C. (2004a). *Owl web ontology language guide* (Inf. Téc.). Disponible en http://www.w3.org/TR/owl-guide/

W3C. (2004b). *Rdf/xml syntax specification* (Inf. Téc.). Autor. Disponible en http://www.w3.org/TR/REC-rdf-syntax/

Wahle, S., y Magedanz, T. (2008). *Network domain federation - an architectural view on how to federate testbeds.* FIREweek.

Watkins, S., y Richardson, M. (2005). *Wp 8: Case study b2b in telecommunications prototype platform design* (Inf. Téc.).

Weijters, A. J. M. M., van der Aalst, W. M. P., y de Medeiros, A. K. A. (2006). Process mining with the heuristicsminer algorithm. *Technology*, *166*, 1-34.

WfMC. (2011). *Xpdl support and resources* (n.º 21/11/2011). Disponible en http://www.wfmc.org/xpdl.html

White, S. (2006). *Introduction to bpmn.* IBM Software Group. Disponible en http://www.zurich.ibm.com/~koe/teaching/ETH2009/White-BPMN-Intro-slides.pdf

Wombacher, A., Fankhauser, P., Mahleko, B., y Neuhold, E. J. (2004). Matchmaking for business processes based on choreographies. En *Ieee international conference on e-technology, e-commerce, and e-services* (p. 359-368). IEEE Computer Society.

Wombacher, A., y Li, C. (2010). Alternative approaches for workflow similarity. En (p. 337-345).

Xu, K., Liu, L., y Wu, C. (2007, Abril). A three-layered method for business processes discovery and its application in manufacturing industry. *Computer Industry*, *58*, 265-278.

Yan, X., y Han, J. (2002). gspan: Graph-based substructure pattern mining. En *Ieee international conference on data mining* (pp. 721–). Washington, DC, USA: IEEE Computer Society.

Yan, X., y Han, J. (2003). Closegraph: mining closed frequent graph patterns. En *9th acm sigkdd international conference on knowledge discovery and data mining* (pp. 286–295). New York, NY, USA: ACM.

Yan, X., Yu, P. S., y Han, J. (2004). Graph indexing: a frequent structure-based approach. En *Acm sigmod international conference on management of data* (pp. 335–346). New York, NY, USA: ACM.

Yan, Z., Cimpian, E., Mazzara, M., y Zaremba, M. (2007, 02/2007). *D1.1 bpmo requirements analysis and design* (Inf. Téc.). SemBiz.

Yan, Z., Dijkman, R. M., y Grefen, P. W. P. J. (2012). Business process model repositories - framework and survey. *Information & Software Technology*, *54*(4), 380-395.

Yun, B., Yan, J., Liu, M., y Yu, Y. (2008). Behavioral equivalence based web service discovery. En *International conference on computer science and software engineering - volume 02* (pp. 368–371). Washington, DC, USA: IEEE Computer Society.

Zaremba, M., Wall, B., y Browne, J. (2002). ebxml registry/repository implementation in agri-food industry. *e-Business Research Forum 2002*.

Zhu, F., Yan, X., Han, J., y Yu, P. S. (2007). gprune: a constraint pushing framework for graph pattern mining. En *11th pacific-asia conference on advances in knowledge discovery and data mining* (pp. 388–400). Berlin, Heidelberg: Springer-Verlag.

Cristhian Figueroa

Ingeniero en Electrónica y Telecomunicaciones (2008), y Magister en Ingeniería Telemática (2012) de la Universidad del Cauca(Colombia). Actualmente es estudiante de Doctorado en Ingeniería Telemática en la Universidad del Cauca y la Universidad Politécnica de Turín (Italia). Además, es investigador del Grupo en Ingeniería Telemática de la Universidad del Cauca, Colombia. Su interés de investigación está centrado en el desarrollo de nuevas técnicas para el descubrimiento de servicios basadas en comportamiento, lingüística (semántica y léxica) y estructura; y su despliegue en entornos web, móvil y de escritorio.

Juan Carlos Corrales

Ingeniero en Electrónica y Telecomunicaciones (1999), y Magister en Ingeniería Telemática (2006) de la Universidad del Cauca (Colombia). Recibió su título de Doctor en Ciencias Especialidad Informática en la Universidad de Versailles Saint-Quentin-en-Yvelines, Francia en 2008. Actualmente, es Profesor Titular y lidera el Grupo de Investigación en Ingeniería Telemática de la Universidad del Cauca. Su interés de investigación está centrado en el desarrollo de nuevas técnicas de recuperación de servicios basados en comportamiento y modelos semánticos; y en la composición automática en plataforma para despliegue de servicios.

Gustavo Ramírez

Ingeniero en Electronica y Telecomunicaciones (2001) y Magister en Ingeniería Telemática (2006) de la Universidad del Cauca. Recibió su titulo de Doctor en Ingeniería Telemática por la Universidad Carlos III de Madrid en 2010. Actualmente es profesor titular e investigador del Departamento de Telemática en la Universidad del Cauca. Ha participado en proyectos nacionales e internacionales en Colombia y España. Sus áreas de interes son la computación móvil y ubicua y los servicios avanzados de telecomunicaciones.