Revista EIA, ISSN 1794-1237 Número 17, p. 105-120. Julio 2012
Escuela de Ingeniería de Antioquia, Medellín (Colombia)

# BUSINESS PROCESS RETRIEVAL BASED ON BEHAVIORAL SEMANTICS

Cristhian Figueroa*
Juan Carlos Corrales**

## ABSTRACT

This paper develops a framework for retrieving business processes considering search requirements based on behavioral semantics properties; it presents a framework called "BeMantics" for retrieving business processes based on structural, linguistics, and behavioral semantics properties. The relevance of the framework is evaluated retrieving business processes from a repository, and collecting a set of relevant business processes manually issued by human judges. The "BeMantics" framework scored high precision values (0.717) but low recall values (0.558), which implies that even when the framework avoided false negatives, it prone to false positives. The highest precision value was scored in the linguistic criterion showing that using semantic inference in the tasks comparison allowed to reduce around 23.6 % the number of false positives. Using semantic inference to compare tasks of business processes can improve the precision; but if the ontologies are from narrow and specific domains, they limit the semantic expressiveness obtained with ontologies from more general domains. Regarding the performance, it can be improved by using a filter phase which indexes business processes taking into account behavioral semantics properties.

KEY WORDS: business process; behavioral semantics; sub-graph isomorphism; control-flow patterns; business process repository.

* Ingeniero en Electrónica y Telecomunicaciones y Magíster en Ingeniería Telemática, Universidad del Cauca; Doctor (c) en Ingeniería Telemática, Universidad del Cauca, y Doctor (c) en Investigación en Ingeniería Informática y de Sistemas, Politecnico di Torino, Italia. Popayán, Colombia. cfigmart@unicauca.edu.co

** Ingeniero en Electrónica y Telecomunicaciones y Magíster en Ingeniería Telemática, Universidad del Cauca; Doctor en Ciencias de la Computación, Université de Versailles Saint-Quentin-en-Yvelines, France. Profesor Titular y líder del Grupo de Ingeniería Telemática (GIT), Universidad del Cauca. Popayán, Colombia. jcorral@unicauca.edu.co

Artículo recibido 16-XI-2011. Aprobado 28-V-2012
Discusión abierta hasta diciembre de 2012

# RECUPERACIÓN DE PROCESOS DE NEGOCIO BASADA EN SEMÁNTICA DEL COMPORTAMIENTO

## RESUMEN

El presente artículo desarrolla un entorno para la recuperación de procesos de negocio teniendo en cuenta requisitos de búsqueda basados en semántica de comportamiento. Presenta un entorno denominado "BeMantics", el cual permite recuperar procesos de negocio basado en propiedades lingüísticas, estructurales y de semántica del comportamiento. La relevancia de este entorno es evaluada recuperando procesos de negocio de un repositorio y reuniendo un conjunto de procesos de negocio relevantes emitidos manualmente por jueces humanos. El entorno "BeMantics" logró valores altos de precisión (0,717), pero valores bajos de exhaustividad (0,558), lo cual implica que aun cuando "BeMantics" evitó falsos positivos, fue propenso a los falsos negativos. El valor de mayor precisión fue logrado para el criterio lingüístico, lo cual demuestra que utilizar inferencia semántica en la comparación de tareas permitió reducir el número de falsos positivos en un factor del 23,6 %. El uso de inferencia semántica en la comparación de las tareas de dos procesos de negocio permite mejorar la precisión. Sin embargo, si las ontologías pertenecen a dominios muy específicos pueden limitar la expresividad obtenida utilizando ontologías de dominios más generales. El rendimiento de BeMantics se puede mejorar empleando una fase de filtro que permita indexar procesos de negocio usando propiedades de semántica de comportamiento.

PALABRAS CLAVE: proceso de negocio; semántica del comportamiento; isomorfismo de grafos; patrones de flujo de control; repositorio de procesos de negocio.

# RECUPERAÇÃO DE PROCESSOS DE NEGÓCIO BASEADA EM SEMÁNTICA DO COMPORTAMENTO

## RESUMO

O presente artigo desenvolve um meio para a recuperação de processos de negócio tendo em conta requisitos de busca baseados em semântica de comportamento. Apresenta um meio denominado "BeMantics", o qual permite recuperar processos de negócio baseado em propriedades linguísticas, estruturais e de semântica do comportamento. A relevância deste meio é avaliada recuperando processos de negócio de um repositório, e reunindo um conjunto de processos de negócio relevantes emitidos manualmente por juízes humanos. O meio "BeMantics" conseguiu valores altos de precisão (0,717), mas valores baixos de exaustividade (0,558), o qual implica que ainda que "BeMantics" evitou falsos positivos, foi propenso aos falsos negativos. O valor de maior precisão foi conseguido para o critério linguístico, o qual demonstra que utilizar inferência semântica na comparação de tarefas permitiu reduzir o número de falsos positivos em um fator do 23,6 %. O uso de inferência semântica na comparação das tarefas de dois processos de negócio permite melhorar a precisão. No entanto, se as ontologias pertencem a domínios muito específicos podem limitar a expressividade obtida utilizando ontologias de domínios mais gerais. O rendimento de BeMantics pode ser melhorado empregando uma fase de filtro que permita indexar processos de negócio usando propriedades de semântica de comportamento.

PALAVRAS-CÓDIGO: processo de negócio; semântica do comportamento; isomorfismo de grafos; padrões de fluxo de controle; repositório de processos de negócio.

## 1. INTRODUCTION

Global economic trends generate highly dynamic markets in which businesses are forced to continuously innovate to improve their competitive position. This situation can be found in horizontal and vertical integration scenarios where ICT (information communications technology) companies (software developers or telecommunications operators) require reusing, creating, adapting, modifying, or integrating existing services reliably to deploy new services. For this reason, currently, ICT companies are looking for new ways to build service-based and flexible business solutions in order to react quickly and cost-effectively to dynamic market conditions.

One way is to adopt flexible technologies based on SOA (Service Oriented Architecture) which provide the capabilities for dynamic composition and easy software components reuse through open standards (Gonçalves da Silva, Ferreira Pires and Van Sinderen., 2011). Those software components can be Web Services (WS) and Business Processes (BP). WS are software units accessible over standard internet protocols, and BP are structures which can integrate other software components (WS or legacy applications with standard interfaces) using a set of connected tasks in order to meet its individual functionalities and achieve a common business purpose (Mongiello and Castelluccia, 2006). However, one of the challenges in this context is to retrieve components within a large repository generated as a consequence of software proliferation. Those retrieved components also have to accomplish acceptable time-to-market and easy reuse properties according the user requirements.

Until now, the software components retrieval methods have been addressed in four discovery levels: interfaces, semantics, structure, and behavior. The first one searches for key words representing names, inputs and outputs of software components (Stroulia and Wang, 2005; Kokash, van den Heuvel and D'Andrea, 2006). The second one, semantics-based discovery uses domain ontologies in order to infer from concepts related to software components names, and concept types of inputs and outputs (Paolucci *et al.*, 2002; Benatallah *et al.*, 2003; Klusch, Fries and Sycara, 2006; Lin and Arpinar, 2006; Choi and Han, 2008; Gonçalves da Silva, Ferreira Pires and Van Sinderen, 2011). The third one, structure-based discovery compares structured software components (i.e. the structure of business processes –BP–) using isomorphism algorithms (Eshuis and Grefen, 2007; Grigori *et al.*, 2010; Wombacher and Li, 2010). And the last one, behavioral-based discovery compares two BP based on their control-flow (i.e. the specific constructors which define the behavior of a BP) (Hidders *et al.*, 2005; Fronk and Lemcke, 2006; Markovic, 2009).

Most methods presented above have applied the discovery levels separately; however, to obtain results with a high degree of reusability and adaption to user requirements, it is necessary to combine them (Sapkota, 2005; Nayak and Lee, 2007; Sellami, Tata and Defude, 2008) and use indexing techniques to accelerate the discovery process. In this context, this paper presents "BeMantics", a framework which addresses BP retrieval from behavioral, structural, and semantic perspectives, and proposes a pre-matching approach called "Behavioral Semantics", which consists of an indexing method based on control-flow patterns (hereafter "patterns") and its semantic relations. Those patterns were introduced by van der Aalst *et al.* (2000) and can be defined as sub-structures which capture a determined behavior from BP and identify the comprehensive BP functionality (Cardoso, 2007).

Furthermore, to evaluate our approach, this paper involved a web platform called "Pertinence Evaluation Tool" (Figueroa, Sandino and Corrales, 2011) which allowed human judges to emit similarity evaluations between a set of 100 BP from real environments (telecommunications and geoprocessing) and a subset of 6 BP acting as queries. The similarity evaluations by the judges obtained in that platform were considered as relevant BP and were used as basis to estimate the precision and recall measures of three automatic BP retrieval tools: the "BeMantics" approach; a structural and lexical tool called "BeMatch" (a platform for matchmaking service behavior models) (Corrales *et al.*, 2008); and an indexing mechanism based on behavioral semantics.

The rest of this paper is structured as follows: the second section presents the framework for BP discovery based on behavioral semantics; the third one describes the materials and methods used to evaluate the "BeMantics" framework; the fourth one shows the results and discussion; and the fifth one exposes the main conclusions of this work.

## 2. BEHAVIORAL SEMANTICS BP RETRIEVAL FRAMEWORK

"BeMantics" (Behavioral Semantics Business Process Retrieval) is a framework to store, match, and retrieve BP based on semantic, structural, and behavioral features. Figure 1 c) and d) shows the "BeMantics" framework architecture which is composed of two main modules: the first one, called Behavioral Semantics BP Repository (hereafter "repository"), is responsible for storing, indexing, and ranking BP according to behavioral semantics characteristics; and the second one, called Structural and Semantics Analyzer (hereafter "structural analyzer"), refines the repository ranking results in order to find an approximate structural and semantic matching between a BP used as reference model called "query BP" and a set of BP stored in the repository called "target BP". The other two modules depicted in figure 1a and 1b, BP Publisher and BP Requestor enable users to design and semantically enrich target and query BP to be stored in the repository and make them available to be retrieved through a matching technique. In this proposal it is used WSMO Studio 0.73 as BP designer to generate semantically enabled XML-based files in the BPMO language (Business Process Modeling Ontology) (Yan *et al.*, 2007).

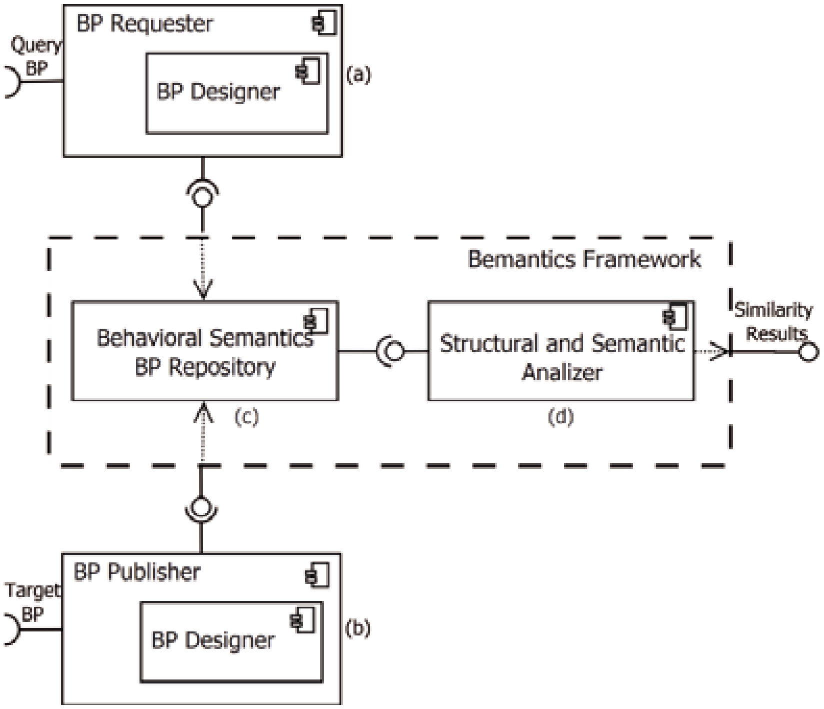


**Figure 1.** Framework for BP retrieval based on behavioral semantics

## 2.1 Behavioral semantics BP repository

The repository is a pre-matching module which offers methods to store, index, and retrieve BP. Its functionality can be viewed as composed of two phases: a storing phase to save target BP, and a retrieving phase which lets users to get a ranked list of stored target BP according to control-flow patterns detected in a query BP. In our proposal, the storing phase starts when the user graphically designs a target BP through the BP designer (figure 1b) obtaining a BPMO process. Subsequently the BPMO process is transformed into a formal model based on graphs, also known as process graph. Then, the repository detects a set of patterns in the target process graph (hereafter "target graph"), labels the BPMO process with those patterns, and stores the target graph, the BPMO process and its detected patterns. In a similar way, the retrieving phase starts when the user graphically designs a query BP as a BPMO process. Subsequently the query BP is transformed into a query process graph (hereafter "query graph"), the repository detects its patterns and then proceeds to find and rank those stored target graphs with a similar set of patterns as the query graph. To facilitate those phases, a base architecture for the repository was designed composed of three main layers: BP parser, patterns analyzer, and storage layer, as can be seen in more detail in our previous work (Rivas *et al.*, 2010).

### *2.1.1 BP parser layer*

Before storing and retrieving a BP it is necessary to transform it to a process graph in order to facilitate the patterns detection and the comparison among BP. The BP parser layer transforms a BPMO to Java objects using the WSMO4J API (Dimitrov *et al.*, 2006), and then applies transformation rules to get a process graph composed of task nodes (events and functions); connector nodes (gateways AND (Split, Join), OR (Split, Join), XOR (Split, Join)) and edges to link those nodes (Corrales, Grigori and Bouzeghoub, 2006). Table 1 describes some connector types defined by the BPMO 1.4 version, as well as the corresponding graph representations used in our approach.

### *2.1.2 Pattern analyzer layer*

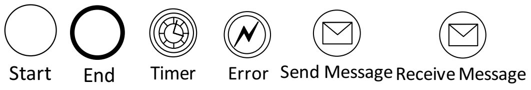
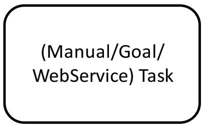
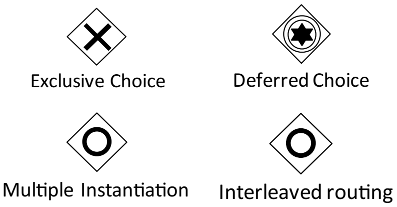
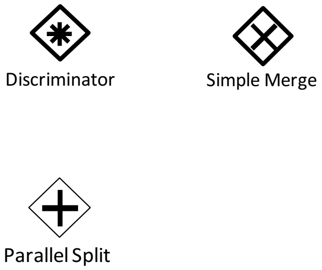
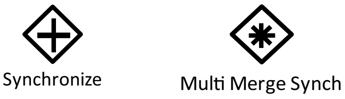
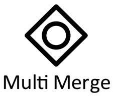
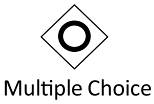
This layer is composed of two sub-layers, a similarity pattern finder and a semantic ranking generator.

***Similarity patterns finder.*** This sub-layer receives as input a process graph and returns its detected patterns set. Those patterns can be represented as sub-graph structures; then the patterns detection problem is converted to a sub-graph isomorphism problem which finds sub-graph structures within a graph (Giugno and Shasha, 2002; Yan, Yu and Han, 2004; Ferro *et al.*, 2007; Zhu *et al.*, 2007).

In this paper, it is used a sub-graph isomorphism approach known as "GraphBlast" (Ferro *et al.*, 2007), which uses an algorithm called VF2 (Cordella *et al.*, 2004) to index sub-structures contained in a large set of graphs (graphs database). This approach uses a graph representation called LNE (List of Nodes and Edges ids format) (Giugno and Shasha, 2002) based on nodes and paths; in this representation the nodes are labeled with a number (*node-id*) and a label (*node-label*), and the paths are lists of node-ids (*id-path*) and node-labels (*label-path*) with unlabeled edges between each two consecutive nodes. Therefore, "GraphBlast" builds an index searching all the paths starting in a determined node and having a predefined length as a query. Additionally, it uses a hash table containing a set of id-paths and id-labels whose keys are the hash values of the label-paths.

Nevertheless, our problem was a little different because it was not required to search a sub-structure in a large set of process graphs, but to search a set of sub-structures in only one process graph at a time. Hence, the "GraphBlast" approach is adapted to our problem using as query a fixed pattern set, with 12 patterns compatible with the BPMO model, and only one process graph where patterns are discovered. In this scenario, each pattern is described using the

**Table 1.** Parsing rules from BPMO rocess to process graph

| BPMO Element | Graph Label | Graph Representation |
|---|---|---|
| **Events**<br><br> Start<br>End<br>Timer<br>Error<br>Send Message<br>Receive Message | **Events**<br>Events are taken as nodes type Event node represented by label (E) | E<br>(Start, End, Timer, Error, SendMessage, ReceiveMessage) Events |
| **Tasks**<br><br> | **Tasks**<br>Tasks are taken as nodes type Task node represented by label (T) | T<br>(Manual, Goal, WebService) Tasks |
| **Gateways**<br><br> Exclusive Choice<br>Deferred Choice<br>Multiple Instantiation<br>Interleaved routing | **Gateways**<br>Gateways are taken as nodes type AND, OR, XOR Split; Join. Represented by labels (ANDS, ANDJ, ORS, ORJ, XORS, XORJ) | XORS<br>(DeferredChoice, ExclusiveChoice, InterleavedParallelRouting, Multiple Instantiation) Gateways |
| <br> | Nodes type XOR Split. Represented by Labels (XORS)<br>Nodes type AND Split. Represented by Labels (ANDS) | XORJ<br>(Discriminator, SimpleMerge) Gateway<br>ANDS<br>ParallelSplit Gateway |
| <br> | Nodes type AND Join. Represented by Labels (ANDJ) | ANDJ<br>(MultipleMergeSynchronise, Synchronisation) Gateways |
| <br> | Nodes type OR Join. Represented by Labels (ORJ) | ORJ<br>Multimerge Gateway |
| <br> | Nodes type OR Split. Represented by Labels (ORS) | ORS<br>MultipleChoice Gateway |

label-path representation, such that index construction was made searching for the number of occurrences of the label-path within a process graph; secondly hash table is defined in order to organize in a matrix with rows with label-paths and column with the graphs stored in the database.

***Semantic ranking generator.*** This sub-layer ranks target graphs according to five distances which represent its numerical differences according to a query graph.

- The *number of patterns distance (Dp)* evaluates the difference between the number of target graph patterns *($P_T$)* and query graph patterns *($P_Q$)*.

$$Dp = \frac{|P_Q - P_T|}{P_Q + P_T} \tag{1}$$

- The *number of nodes distance (Dn)* evaluates the distance according to the number of control-flow nodes (ANDS, XORJ, XORS, etc.) in the target graph (Nt) and the query graph (Nq).

$$Dn = \frac{|N_q - N_t|}{N_q + N_t} \tag{2}$$

- The third one, the *number of edges distances (De)*, evaluates the number of edges in the target graph (Et) and the query graph (Eq).

$$De = \frac{|E_q - E_t|}{E_q + E_t} \tag{3}$$

- The *route-descriptor distance ($Drd_T$)* evaluates the distance in terms of number of occurrences (oNq and oNt) and position of each pattern detected in the target graph and the query graph (PPq and PPt). In this case, the total route-descriptor distance is the sum of all route-descriptor distances of each pattern detected in both graphs.

$$Drd_T = \sum \frac{\frac{|PP_q - PP_t|}{PP_q + PP_t}}{|oN_q - oN_t|} \tag{4}$$

For example, suppose to have two process graphs *$BP_1$* and *$BP_2$* which share two patterns XORS and XORJ. The pattern occurrences and positions are described as *$PBP_i$* = {Pattern$_j$ (k)}, where *$PBP_i$* is the set of patterns detected in a *$BP_i$*; *j* is a pattern occurrence, and *k* the pattern occurrence position in the BP. In this way, for *$BP_1$* and *$BP_2$* can be stated the next pattern occurrences and positions: *$PBP_1$* = {*$XORJ_1$* (6), *$XORS_2$* (9)}, and *$PBP_2$* = {*$XORS_1$* (2), *$XORJ_2$* (4), *$XORS_3$* (7), *$XORS_4$* (10)} respectively.

Therefore, *Drd* for the first pattern (XORJ) can be calculated as

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

and for the second pattern (ORS) as.

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

Finally, the total route-descriptor distance:.

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

- The *semantic patterns distance Dsp* estimates the difference between two patterns according to the general pattern relationships defined by Russell *et al.* (2006). To do that a relationship abstraction between the 12 patterns previously selected called *"control-flow patterns ontology"* was created. This ontology has two main relationships of patterns: specialization, when one pattern has a more restricted view than the other; and composition, when one pattern can be represented as the union of other patterns. Therefore, the ontology and the relationships were used in order to find a semantic distance between patterns contained in process graphs. This semantic distance was obtained calculating a leap distance ($D_{leap}$) (Ge and Qiu, 2008) between two patterns in the ontology:

$$\mathrm{D_{leap}} = 1 + \frac{1}{2^{\mathrm{depth}}} \tag{5}$$

In equation 5 "*depth*" is the number of leaps in the ontology from root to the goal concept. In this way, the total semantic distance is the sum of all values over each path between each concepts couple:

$$Dsp = \sum \mathrm{D}_{\mathrm{leap}} \tag{6}$$

Finally, a total patterns distance (DpT) between a query and a target graph is calculated as the percentage sum of the five distances presented before.

$$DpT = Dp + Dn + De + Drd_T + Dsp \tag{7}$$

In this regard, the ranking of target graphs is organized according to the distance values in an ascending order with respect to the query graph.

### 2.1.3 Storage layer

This layer contains three sub-repositories. The first one is responsible for storing BP models represented as WSML documents. The second one called BP Graphs uses a graph database to store: the graphs representing the patterns, the query BP, and the target BP. And the third one uses a relational database (RDBMS) which contains BP references to the WSML documents location and its references with the patterns and graph representations.

## 2.2 Structural and semantic analyzer

This second module of the "BeMantics" Framework was designed in order to execute a structural matching which modifies each target graph obtained from the repository (pre-ranked target graphs) in order to make it as similar as possible to a query graph. Each modification performed by the structural matchmaking is called an edit operation; therefore, the more edit operations are made, the more different is the target graph with respect to the query graph. The result of these edit operations is called a result graph, which reflects the changes applied to the target graph. Finally, the result graph is evaluated in terms of structure, linguistics, and sequential behavior, in order to estimate the difference between the query graph and the target graph. Figure 2 shows the Structural and Semantic Analyzer components which are described next.

### 2.2.1 Structural matchmaker

The structural matchmaker (figure 2a) receives a query graph as input and structurally compares it with each graph of a set of pre-ranked target graphs (i.e. those target graphs ranked by the repository as was presented in the section 2.1). The structural comparison is based on the error-correcting subgraph isomorphism algorithm (Messmer, 1995) previously implemented by Grigori *et al.* (2010), which starts creating a set of mappings from each node of the query graph to each node in the target graph. Each mapping represents a set of edit-operations (modifications in the target graph as delete, insert, or substitute nodes and edges). Then, the algorithm calculates an edit-cost for each edit-operation (i.e. the cost of performing edit-operations in the target graph). Finally, the algorithm uses the edit-cost of each edit-operation to estimate a total cost for each mapping, and concludes either when a minimal mapping (a mapping with a minimal edit-cost) is found or when all the possible mappings have exceeded a given acceptance cost (*AC*) predefined by the user. In our proposal, the edit-cost and the total cost were calculated by the cost function builder module (figure 2c).

### 2.2.2 Cost function builder

This module (figure 2c) calculates the cost functions for the edit-operations in order to estimate the distance between a query graph and a target graph. In our proposal the process graphs are composed of task nodes, representing business activities of the BP; connectors, expressing the control-flow constraints; and the edges forming links among nodes. Therefore, a set of valid edit-operations are defined to be executed by the structural matchmaker;

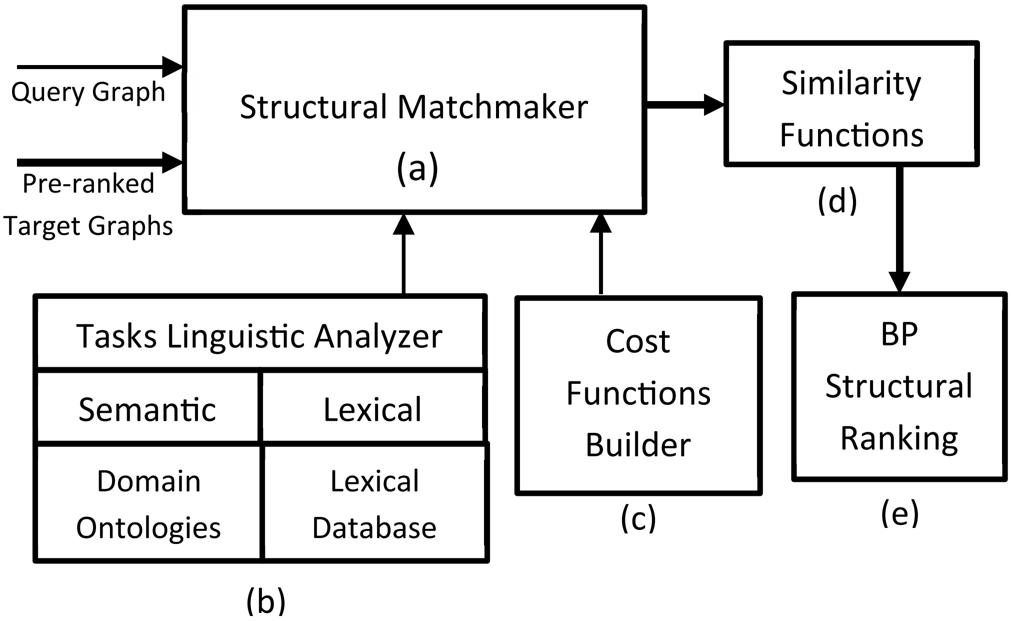


**Figure 2.** Structural and Semantic Matchmaker

those operations are substitute or delete task nodes, insert or delete connectors, and insert or delete edges.

For this reason, the costs for these edit-operations are calculated using the edge distance, connector distance, task node distance, and total distance. Besides, multiplication factors are defined to allow users to customize restrictions to the matchmaker with the purpose to execute one edit-operation with more possibility than others. For example, if one user sets the substitute node value to 0.2 and the delete node value to 0.5, then the algorithm will prefer to edit the node rather than removing it.

*Edge distance (ed)*. This function measures the number of deleted (*de*) and inserted edges (*ie*) in the target graph and multiplies them by a deleted edge factor ($\vartheta$) and an inserted edge factor ($\mu$) predefined by the user.

$$ed = \vartheta \sum de + \mu \sum ie \tag{8}$$

*Connector distance (cd)*. This function measures the number of deleted and inserted connectors in the target graph and multiplies them by a deleted connector factor ($\rho$) and an inserted connector factor ($\sigma$) predefined by the user.

$$cd = \rho \sum dc + \sigma \sum ic \tag{9}$$

*Task node distance (tnd)*. This function measures the number of deleted task nodes (*dn*) or evaluates the cost of substitute a query node by a target node (*sn*) (i.e. finding a task node in the target graph which can be assigned to the mapping as replacing node for a specific query node due to its similarities in terms of task name and interfaces (inputs/outputs)). The substitute operation is estimated by the linguistic analyzer (figure 2b) which is described in section 2.2.3. Like the other distances, this function also multiplies the deleted nodes and the substitute costs for a deleted node factor ($\tau$) and a substitute node factor ($\varphi$) predefined by the user.

$$tnd = \tau \sum dn + \varphi \sum sn \qquad (10)$$

*Total distance (TD):* the total distance sums the edge, connector, and task node distances with the purpose to give a final total distance among the BP.

$$TD = ed + cd + tnd \qquad (11)$$

### 2.2.3 Linguistic analyzer

This module (Figure 2b) evaluates the substitute operation by comparing task nodes in two process graphs. In previous works (Corrales *et al.*, 2008) this comparison was executed using lexical analyzers which compare strings of the task node names. In our proposal the BPMO model allowed to classify the task nodes as event nodes, to represent a task executed according to stimulus (e.g. an alarm, an error or a time deadline); and function nodes to represent BP activities with name and interfaces (inputs/outputs) capable to be semantically enriched. Hence, to evaluate the substitute-operation for function nodes a semantic analyzer is used, and for the event nodes, a lexical analyzer. Next are described both analyzers.

#### Lexical analyzer

The lexical distance evaluates the difference between the names of two nodes, analyzing combinations of words and abbreviations. In this proposal existing algorithms are used to find the lexical distance: the NGram, Check synonym, and Check abbreviation. The NGram algorithm estimates the similarity according to a number of common sequences of defined-length characters (q-grams) between the node names; the Check abbreviation uses a custom abbreviation dictionary, and the Check synonym algorithm finds its synonyms using a lexical database (in our case WordNet was selected as lexical database (Miller, 1995)). In our implementation the equation proposed by Patil *et al.* (2004) is applied, which uses the results of the NGram (*m1*), Check synonym (*m2*) and Check abbreviation (*m3*) algorithms to evaluate the lexical distance (LS) between two nodes names.

$$LS = \begin{cases} 1 & if\ (m1{=}1 \vee m2{=}1 \vee m3{=}1) \\ m2 & if\ (0 < m2 < 1 \wedge m1{=}\ m3{=}0) \\ 0 & if\ (m1{=}\ m2{=}\ m3{=}0) \\ \frac{m1+m2+m3}{3} & if\ m1,m2,m3 \in (0,1) \end{cases} \qquad (12)$$

#### Semantic analyzer

The semantic analyzer calculates a substitute-cost between two function nodes. In this case the substitute-cost is called a semantic distance and can only be calculated in function of nodes semantically enriched, it means function nodes with names and interfaces having semantic annotations related to concepts from domain ontologies. Domain ontologies have a concept tree from a specific domain, where the semantic distance is obtained calculating a leap distance ($D_{leap}$) (Ge and Qiu, 2008) between two concepts (*c1, c2*) in the ontology:

$$\mathrm{D}_{\mathrm{leap}}(c_1, c_2) = 1 + \frac{1}{2^{\mathrm{depth}}} \qquad (13)$$

In this equation depth is the number of leaps in the ontology from root to a goal concept, hence the total semantic distance (SD) is the sum of all values over each path between each concepts couple:

$$\mathrm{SD} = \sum \mathrm{D}_{\mathrm{leap}}(c_i, c_j) \qquad (14)$$

In this way, the names and interfaces of function nodes are compared according the semantic distance. However, to calculate this distance in interfaces (inputs/outputs), the algorithm looks for a task combination which "best cover" the interfaces of the query graph (i.e. to determine a target task *TT* from inputs and outputs of a query task *TQ*, such that *TT* shares the large possible number of outs with *TQ* and without exceed the inputs of *TQ*).

In our proposal the function name and its interfaces are semantically enriched by using two domain ontologies from telecommunications domain: seTOM (enhanced Telecom Operations Map) (Shangguan, Gao and Zhu, 2007) to enrich the names and sSID (shared information and data model) (Liu, Lv and Kang, 2006) to enrich the interfaces. Both ontologies are part of ontologies framework YATOSP (Yet Another Telecoms Ontology, Service and Process) (Martínez and Pérez, 2008).

### 2.2.4 Similarity results

This module (figure 2d) evaluates the similarity between a query graph and a target graph using the edit-costs presented in previous sections. The similarity functions are classified in three BP properties: structure, linguistics, and sequential behavior.

*Structural Similarity (Stsim)*. It computes the similarity as function of the total edit-distance (*TD*) found by the structural matching algorithm.

$$Stsim = \frac{1}{1+TD} \tag{15}$$

*Linguistic Node Similarity (LNsim)*. It returns a similarity value computed as function of the structural similarity, the number of nodes in the result graph corresponding to the query graph (*intersected nodes*), and the total number of nodes in the query graph (*query nodes*) according to the linguistic comparison presented in section 2.2.3.

$$LNsim = \frac{Stsim * intersected\ nodes}{query\ nodes} \tag{16}$$

*Sequential Behavior Similarity (SBsim)*. It estimates the similarity as function of the structural similarity and the number of n-sequences (sequences composed by *n* nodes). This similarity measure relates the n-sequences in the query (*nSeqQ*), target (*nSeqT*), and result (*nSeqT*) graphs.

$$SBsim = \frac{Stsim}{1+nSeqQ+nSeqT-2*nSeqT} \tag{17}$$

Using these three similarity measures, the target BP are classified in three ranks from the most similar to the less similar respect to a query BP. Those ranks are finally presented to the user by the BP Structural Ranking module (figure 2e), which has a graphical user interface allowing users to visualize the similarity results, executed edit-operations and the result graph matched with the query graph.

## 3. MATERIALS AND METHODS

This section describes the materials and methods used in this proposal to experimentally test the Behavioral Semantics BP Retrieval Framework.

### 3.1 A BP test set

To test the "BeMantics" framework, a BP test set was created with 60 BP from the telecommunications domain and 40 BP from the geoprocessing domain modeled using the BPMO language. Only, the telecommunications BP were semantically enriched by using the domain ontologies seTOM y sSID.

### 3.2 A pertinence evaluation model

A pertinence evaluation model (Figueroa, Sandino and Corrales, 2011) was designed in order to allow a set of 6 human judges to issue relevance judgments for the BP of the test set and 6 BP acting as queries. Therefore, this model is useful as it allows to catalog different discovery BP tools by their level of retrieval effectiveness. If BP tools and evaluators come up with a similar result, effectiveness is given.

#### 3.2.1 Manual criteria

The manual criteria represent BP properties which facilitate the evaluation of the judges. In this paper four criteria levels commonly analyzed in BP discovery tools (Bernstein *et al.*, 2005; Goderis *et al.*, 2009; Wombacher and Li, 2010) are selected. The first one, the structure criterion represents the graphical structure and the dependence relationships (causal dependence) of the tasks into two BPs. The second one, the linguistics level denotes the semantic and lexical features of the tasks name, descriptions, and interfaces. And the last one, the behavioral level is the control-flow of BP.

### 3.2.2 BP recovery measures

The recovery measures evaluate the criteria in terms of recovery performance and relevance (Blair, 1990; Baeza-Yates and Ribeiro-Neto, 1999; Borlund, 2000; Pors, 2000). The performance is related with response time; and the relevance with recovery effectiveness, which can be estimated by using precision (P) and recall (R) measures. Precision evaluates the system's ability to recover only relevant elements (those elements considered as similar to a query by the judges) and avoid unexpected results or false positives (i.e. retrieved non-relevant elements). And the recall evaluates the ability to recover all the relevant elements avoiding missing relevant results (i.e. false negatives).

In this proposal a relevance model (Pg y Rg) (Küster and König-Ries, 2008) is presented; it considers diverse levels of relevance in contrast with the traditional binary relevance which only consider two levels (relevant or not relevant) (Baeza-Yates and Ribeiro-Neto, 1999). In this way the recovery relevance of our framework is estimated by comparing a query BP ($Q$) with each element ($T_i$) of a set of target BP using the "BeMantics" framework to get an automatic ranking ($f_2$) and the "Pertinence Evaluation Tool" to get the real ranking ($f_r$) generated by the judges evaluation. The Pg and Rg is calculated using the equations 18 and 19.

$$Pg = \frac{\sum_{T_i \epsilon T} \min\{f_r(Q,T_i), f_e(Q,T_i)\}}{\sum_{T_i \epsilon T} f_e(Q,T_i)} \qquad (18)$$

$$Rg = \frac{\sum_{T_i \epsilon T} \min\{f_r(Q,T_i), f_e(Q,T_i)\}}{\sum_{T_i \epsilon T} f_r(Q,T_i)} \qquad (19)$$

## 4. RESULTS AND DISCUSSION

This section discusses the experimental evaluation and the obtained results for the "BeMantics" prototype. The experimental evaluation was executed in a test server with next features: 4GB of RAM, an Intel i3-530 (2.93 GHz) processor and a Linux Ubuntu 10.04 operating system. Moreover, to study experimentally the "BeMantics" prototype two analyses were designed: the first one to evaluate the performance of the prototype (section 4.1), and the second one to estimate its relevance (section 4.2). Those analyses were executed in the two main modules of "BeMantics", the repository and the structural analyzer.

## 4.1 Performance analysis

### 4.1.1 Behavioral Semantics BP Repository

To analyze the repository performance a set of 100 BP was stored and used to test the ranking system with 6 queries (query BP). The results showed the time consumption takes values in the range of 10 ms to 40 ms generating rankings composed of 40 to 90 BPs.

### 4.1.2 Structural and Semantic Analyzer

In this analysis the execution parameters (section 2.2.1) were arbitrarily selected as next: 0.5 for delete edge ($\vartheta$), insert edge ($\mu$), delete connectors ($\rho$) and insert connectors ($\sigma$); 1.0 for delete nodes ($\tau$); and 0.7 for substitute nodes ($\varphi$). The execution parameter acceptance cost (AC), which determines the maximum cost of the edit-operations, was experimentally tested, and it was found that with the value set to 5, "BeMantics" achieved an intermediate time consumption and a good matching quantity, because for superior values the time consumption was exaggerated or the virtual memory overloaded; on the other hand, for inferior values a considerable reduction in the matching quantity was observed.

Next, having fixed the AC value to 5, the structural analyzer performance was analyzed by using the average nodes-number for the entire target BP set and the matching average-time consumption for each matching, as can be seen in figure 3.

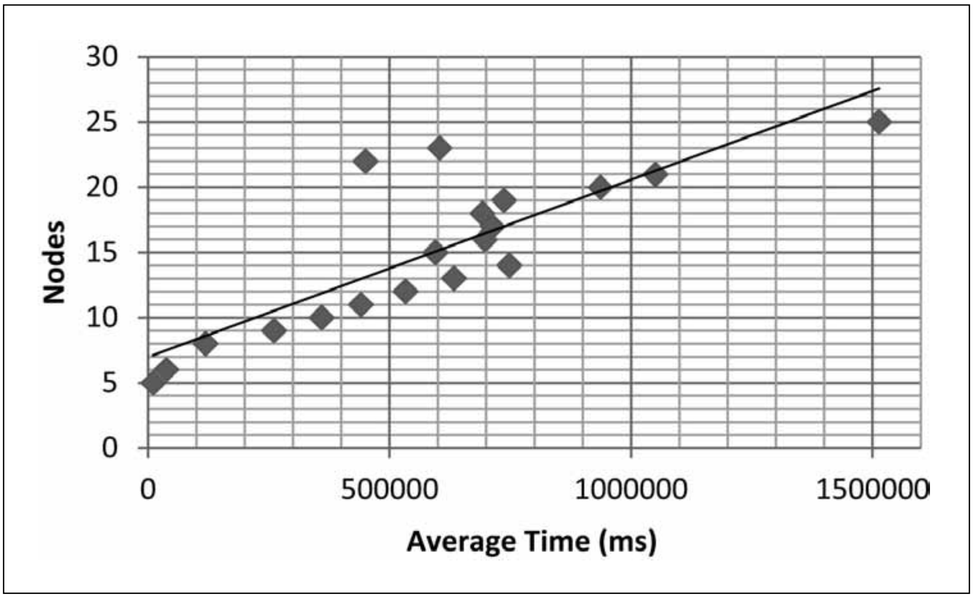


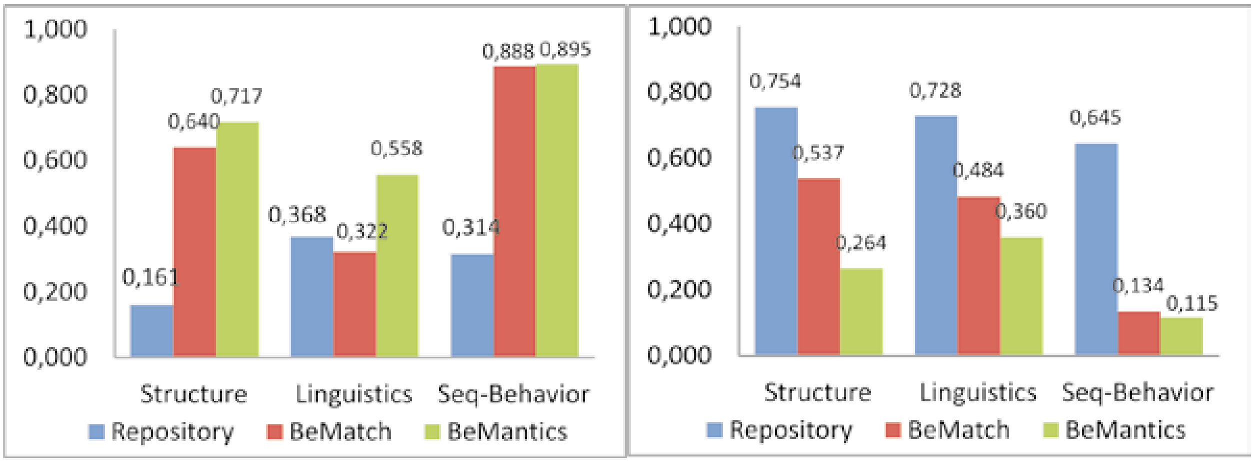
**Figure 3.** Matching time consumption vs. BP nodes

Figure 3 shows that the time consumption, for matching BPs with a range of nodes between 5 and 30, presents a linear behavior as the nodes number increase in each graph. In conclusion, the time consumption depends more on the parameter selection than the number of nodes in the BP.

## 4.2 Retrieval relevance analysis

This analysis was executed to determine the average graded measures for precision (Pg) and recall (Rg) for three BPs retrieval prototypes according structural, linguistics, and sequential behavior criteria, as shown in figure 4. The first one, called repository, is the Behavioral Semantics Repository (section 2.1) which retrieves BP using the control-flow patterns; the second one, called “BeMatch”, retrieves BP using structural and lexical criteria, and the third one is the “BeMantics” prototype (section 3.2) which is an improved version of “BeMatch”, using semantic instead of the lexical criterion and a pre-indexing featured repository.

Comparing the graded precision (figure 4a), “BeMantics” scored the highest values (0.717, 0.558 and 0,895) in the three criteria, followed by “BeMatch” (0.640, 0.322 and 0,888) and the repository with the lowest values (0.161, 0.368 and 0.314). This implies that “BeMantics” was the most accurate prototype for retrieving BPs. Additionally, “BeMantics” improved the linguistics criterion of “Bematch” by 23.6 %, showing that the use of semantics inference to compare tasks can reduce the number of false positives by 23.6 % (i.e. not relevant BP retrieved). Nevertheless, comparing the graded recall (figure 4b), “BeMantics” scored the lowest Rg (0.264, 0.360 and 0.115) in all the criteria, followed by “BeMatch” (0.537, 0.484 and 0.134) and the repository with the highest values (0.754, 0.728 and 0.645). This means that “BeMantics” was more likely to generate false negatives (i.e. relevant BP not retrieved); this could be produced because some relevant processes were lost in the ranking phase by the repository, even when it showed high Rg values.



(a) Average graded precision (Pg)

(b) Average graded recall (Rg)

**Figure 4.** Average graded precision and recall for the repository, “BeMantics” and “BeMatch”

To finalize, the results for repository showed that it scored the best Rg and the worst Pg, which is normal, because the repository was designed as a pre-matching ranking system and therefore its functionality is to create an initial filter with a high level of Rg (lower level of false negatives), and as a filter its Pg is not so important due to it is supposed is a matter of the matching algorithms which has to improve the Pg, as can be seen in figure 4b.

## 5. CONCLUSIONS

This paper presents “BeMantics”, a framework which addresses BP retrieval from behavioral, structural, and semantic perspectives. “BeMantics” was designed with two main modules; first a behavioral semantics pre-matching repository to allow storing and retrieving BP; and second a structural and semantic matchmaker which refines the repository results using an error-correction isomorphism algorithm.

To analyze the “BeMantics” framework, the performance and relevance analysis are executed for the repository and the structural analyzer. The performance evaluation showed that the repository presented lower time consumption (10-40 ms) and therefore if it were applied as pre-matching phase allowed to reduce the search space and the total time consumption for the “BeMantics” framework which presented a poor performance. The relevance evaluation was executed comparing “BeMantics” and a previous work called “BeMatch” with a similar structural analyzer. This evaluation showed that the semantics inference of “BeMatch” can increase the graded precision, but reduces the graded recall regarding “BeMatch”, which only executes a lexical comparison and therefore “BeMantics” was more accurate than “BeMatch”, but lost some relevant processes due the pre-matching phase. In conclusion, it can be considered that using ontologies from narrow and specific domains (e.g. telecommunications) limits the semantic expressiveness that can be obtained by using ontologies from more general domains, because the set of concepts used to enrich activities and interfaces is reduced to the concepts set included in the domain ontology. Consequently, this paper recommends the use of more general ontologies because they offer a more open concept set; nevertheless, if the execution context of the BP is well-defined, using narrow specific domain ontologies can reduce ambiguities between professionals who enrich the BP.

Finally, in order to improve the “BeMantics” framework the next steps of the research are planned as follows: first to add more criteria such as BP nonfunctional properties (Guerrero, Corrales and Raggia, 2010), and second to use heuristics and mining methods based on historical execution data of BP (van der Aalst *et al.,* 2010).

## ACKNOLEDGMENTS

The authors would like to thank University of Cauca, Colciencias and COOPEN for supporting the Ph.D. student Cristhian Figueroa.

## REFERENCES

Baeza-Yates, R. A. and Ribeiro-Neto, B. *Modern information retrieval*. Addison-Wesley Longman, 1999.

Benatallah, B., Hacid, M. S., Rey, C. and Toumani, F. (2003). *Semantic reasoning for web services discovery*. Proceedings of WWW Workshop on E-Services and the Semantic Web. Budapest.

Bernstein, A.; Kaufmann, E.; Bürki, C. and Klein, M. *How similar is it? Towards personalized similarity measures in ontologies*. 7th International Conference Wirtschaftsinformatik. Part 15, pp.1347-1366. Bamberg, Germany: Physica, 2005.

Blair, D. C. *Language and representation in information retrieval*. Elsevier North-Holland, 1990.

Borlund, P. (2000). “Experimental components for the evaluation of interactive information retrieval systems”. *Journal of Documentation,* vol. 56, No. 1 (January), pp. 71-90.

Cardoso, J. *Business process quality metrics: Log-based complexity of workflow patterns. In: On the Move to Meaningful Internet Systems*. Meersman, R. and Tari, Z. (eds.). Part I. LNSC, vol. 4803. Springer Berlin / Heidelberg, 2007. Pp. 427-434.

Choi, O. and Han, S. (2008). Flexible rule-based web services system for users' preferences. 4th *International Conference on Next Generation Web Services Practices,* (20-22 October), pp. 1-4.

Cordella, L. P.; Foggia, P.; Sansone, C. and Vento, M. (2004). “A (sub)graph isomorphism algorithm for matching large graphs”. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, No. 10 (October), pp. 1367-1372.

Corrales, J. C.; Grigori, D. and Bouzeghoub, M. (2006). *BPEL Processes matchmaking for service discovery*. Proceedings of Cooperative Information Systems.

Corrales, J. C.; Grigori, D.; Bouzeghoub, M. and Burbano, J. E. *BeMatch: A platform for matchmaking service behavior models*. 11th International Conference on Extending Database Technology, 2008.

Dimitrov, M.; Momtchev, V.; Simov, A.; Ognyanoff, D. and Konstantinov, M. (2006). *wsmo4j Programmers guide v. 2.0.1* [Online]. OntoText Lab. [consulted on April 24, 2012]. Available in: http://wsmo4j.sourceforge.net/doc/wsmo4j-prog-guide.pdf

Eshuis, R. and Grefen, P. W. P. J. *Structural matching of BPEL processes*. European Conference on Web Services. IEEE Computer Society, 2007.

Ferro, A.; Giugno, R.; Mongiovi, M.; Pulvirenti, A.; Skripin, D. and Shasha, D. (2007). *GraphBlast: Multi-feature graphs database searching*. Workshop on Network Tools and Applications in Biology (NETTAB), Pisa, Italy (12-15 June).

Figueroa, C.; Sandino, L. y Corrales, J. C. (2011). “Plataforma para evaluar sistemas de recuperación de procesos de negocio”. *Revista de Investigaciones UCM,* vol. 17, No. 17 (mayo), pp. 64-76.

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

Ge, J. and Qiu, Y. (2008). *Concept similarity matching based on semantic distance*. Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid. IEEE Computer Society.

Giugno, R. and Shasha, D. (2002). *GraphGrep: A fast and universal method for querying graphs*. Proceedings of 16th International Conference on Pattern Recognition, 2002, Quebec City, Canada (11-15 August), vol. 2, pp. 112-115.

Goderis, A.; Fisher, P.; Gibson, A.; Tanoh, F.; Wolstencroft, K.; De Roure, D. and Goble, C. (2009). “Benchmarking workflow discovery: A case study from bioinformatics”. *Concurrency and Computation: Practice and Experience,* vol. 21, No. 16 (November), pp. 2052-2069.

Gonçalves da Silva, E.; Ferreira Pires, L. and Van Sinderen, M. (2011). “Towards runtime discovery, selection and composition of semantic services”. *Computer Communications*, vol. 34, No. 2 (February), pp. 159-168.

Grigori, D.; Corrales, J. C.; Bouzeghoub, M. and Gater, A. (2010). “Ranking BPEL processes for service discovery”. *IEEE Transactions on Services Computing,* vol. 3, No. 3 (July-September), pp. 178-192.

Guerrero, E.; Corrales, J. C. y Raggia, R. (2010). *Recuperación de servicios basada en la personalización del proceso de descubrimiento*. Proceedings of Euro-American Conference on Telematics and Information Systems. Panama City.

Hidders, J.; Dumas, M.; van der Aalst, W. M. P.; Ter Hofstede, A. H. M. and Verelst, J. *When are two workflows the same? CATS*. Australian Computer Society, 2005.

Klusch, M.; Fries, B. and Sycara, K. (2006). *Automated semantic web service discovery with OWLS-MX*. Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. Hakodate, Japan: ACM.

Kokash, N.; van den Heuvel, W.-J. and D'Andrea, V. (2006). *Leveraging web services discovery with customizable hybrid matching*. Proceedings of International Conference on Service Oriented Computing.

Küster, U. and König-Ries, B. (2008). *On the empirical evaluation of semantic web service approaches: Towards common SWS test collections*. Proceedings of the 2008 IEEE International Conference on Semantic Computing. IEEE Computer Society.

Lin, L. and Arpinar, I. B. (2006). *Discovery of semantic relations between web services*. IEEE International Conference on Web Services (ICWS 2006) Chicago, IL (18-22 September), pp. 357-364.

Liu, Y.; Lv, W. and Kang, J. *Towards the NGOSS SID ontology based on description logics*. *GLOBECOM*. IEEE, 2006.

Markovic, I. (2009). *Enhanced process query framework*. Germany patent application.

Martínez, J. y Pérez, N. *YATOSP: Marco de referencia semántico para el sector Telco*. *Telecom I+D Bilbao 08*. Bilbao, Spain, 2008.

Messmer, B. (1995). *Graph matching algorithms and applications*. University of Bern.

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM,* vol. 38, No. 11 (November), pp. 39-41.

Mongiello, M. and Castelluccia, D. 2006. *Modelling and verification of BPEL business processes*. Proceedings of Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MBD/MOMPES'06), pp. 144-148.

Nayak, R. and Lee, B. (2007). *Web service discovery with additional semantics and clustering*. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society.

Paolucci, M.; Kawamura, T.; Payne, T. R. and Sycara, K. (2002). *Semantic matching of web services capabilities*. Proceedings of the First International Semantic Web Conference.

Patil, A.; Oundhakar, S.; Sheth, A. and Verna, K. METEOR-S web service annotation framework. *13th International Conference on the World Wide Web*, 2004.

Pors, N. O. (2000). "Information retrieval, experimental models and statistical analysis". *Journal of Documentation,* vol. 56, No. 1, pp. 55-70.

Rivas, D. F.; Corchuelo, D. S.; Figueroa, C. and Corrales, J. C. (2010). *Business process repository based on control flow patterns*. Proceedings of Euro-American Association on Telematics and Information Systems. Panama City (22-24 September), 4 p.

Russell, N.; ter Hofstede, A. H. M.; Aalst, W. M. P. and Mulyar, N. *Workflow control-flow patterns: A revised view*. In: BPM Center.org (ed.) BPM Center Report BPM-06-22 BPM Center, 2006.

Sapkota, B. (2005). Web service discovery in distributed and heterogeneous environment. Procedings of the *WWW Service Composition with Semantic Web Services Workshop, Compiegne, France* (19 September).

Sellami, M.; Tata, S. and Defude, B. *Service discovery in ubiquitous environments: Approaches and requirements for context-awareness*. Business Process Management Workshops. Springer, 2008.

Shangguan, Z.; Gao, Z. and Zhu, K. *Ontology-based process modeling using eTOM and ITIL*. International Conference on Research and Practical Issues of Enterprise Information Systems. Vol. 2. Boston: Springer, 2007.

Stroulia, E. and Wang, Y. (2005). "Structural and semantic matching for assessing web-service similarity". *International Journal of Cooperative Information Systems*.

Van der Aalst, W. M. P.; Barros, A. P.; ter Hofstedey, A. H. M. and Kiepuszewski, B. (2000). Advanced workflow patterns. *Cooperative Information Systems*. Springer.

Van der Aalst, W. M. P.; Rubin, V.; Verbeek, H. M. W.; van Dongen, B. F.; Kindler, E. and Günther, C. W. (2010). "Process mining: A two-step approach to balance between underfitting and overfitting". *Software and System Modeling,* vol. 9, No. 1 (January), pp. 87-111.

Wombacher, A. and Li, C. (2010). *Alternative approaches for workflow similarity*. 2010 IEEE International Conference on Services Computing, pp. 337-345.

Yan, Z.; Cimpian, E.; Zaremba, M. and Mazzara, M. (2007). *BPMO: Semantic business process modeling and WSMO extension*. IEEE International Conference on Web Services. Salt Lake City, UT (9-13 July), pp. 1185-1186.

Yan, X.; Yu, P. S. and Han, J. (2004). *Graph indexing: a frequent structure-based approach*. Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France: ACM.

Zhu, F.; Yan, X.; Han, J. and Yu, P. "gPrune: A constraint pushing framework for graph pattern mining". In: Advances in Knowledge Discovery and Data Mining Zhou, Z.-H.; Li, H. and Yang, Q. (eds.). Berlin/Heidelberg: Springer, 2007.