S-Discovery: A Behavioral and Quality-based Service Discovery on the Cloud

Ahmed Gater[1], Fernando Lemos[2], Daniela Grigori[1] and Mokrane Bouzeghoub[2]
[1]*Université Paris-Dauphine, Pl. Mal de Lattre de Tassigny, 75775 Paris, France*
[2]*Université de Versailles Saint-Quentin en Yvelines, 45 Avenue des Etats-Unis, 78035 Versailles Cedex, France*

Keywords: Business Process as a Service, Service Oriented Architecture, Service Discovery, Business Process Matching.

Abstract: Cloud computing recently emerged as a paradigm providing computer power and storage as a utility that is consumed on demand (following the footsteps of other utilities, like electricity). Recently, a new service delivery mode emerged: Business Process as a Service (BPaaS). As a consequence, process models repositories will be developed allowing this new type of services to be published by process providers and discovered by enterprises wanting to outsource some of, or parts of, their processes. In this paper we present the S-Discovery framework allowing to find in such repositories processes that could satisfy user functional and no-functional requirements.

1 INTRODUCTION

Cloud computing emerged recently as a paradigm providing computer power and storage as a utility that is consumed on demand (following the footsteps of other utilities, like electricity). It allows typically three delivery modes: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

The development of cloud computing offers new opportunities for enterprises to outsource their processes and thus a new service delivery mode emerged: Business Process as a Service (BPaaS) (Anstett et al., 2009; Pathirage et al., 2011). As a consequence, process models repositories will be developed allowing this new type of services to be published by process providers and discovered by enterprises wanting to outsource some of, or parts of, their processes. Similar to service registries, process repositories will contain process model descriptions, but also business specifications (non-functional descriptions like cost, quality of service, etc). We argue that process model discovery techniques taking into account functional and non-functional requirements will be required.

Besides business processes outsourcing, other application scenario for BPaaS can be found in the area of scientific workflows (Pathirage et al., 2011). Many large scale collaborative science projects use workflows to automate computation steps. However, defining and running such workflow systems are often a challenge. Workflow engines in the cloud would facilitate scientist's work and reduce overhead on their projects. In the context of web-based scientific workflow repositories, scientists expressed the need to have workflow similarity search capabilities (Goderis et al., 2006). Moreover, in the new context of BPaaS repositories, they may be interested in finding, among the retrieved workflows the one satisfying some business criteria (e.g., cost) and some global or local quality requirements, e.g., the workflow that takes the shorter time or which guarantees a given correctness for the results of a specific activity.

These scenarios show the need for a discovery approach taking into account both process model and non-functional requirements. The contribution of this paper is a framework able to efficiently query a repository to find processes that could best fulfill user structural and non-functional requirements. We suppose that users express their query as a process model accompanied by non-functional requirements. Thus, the technical challenges in the discovery approach are at two levels. At the description level, (i) provide a formal model that allows one to specify, at different granularity levels, non-functional attributes as annotations of the functional specification; and (ii) allow the user to enrich his query with non-functional requirements. At the discovery level, (i) filter the repository to efficiently find matching candidates (ii)

Gater A., Lemos F., Grigori D. and Bouzeghoub M..
S-Discovery: A Behavioral and Quality-based Service Discovery on the Cloud.
DOI: 10.5220/0004378201040109
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 104-109
ISBN: 978-989-8565-52-5

combine the structure-based matching algorithms and non-functional factors matching and (iii) define a similarity measure that aggregates both functional and non-functional similarities.

The remainder of the paper is structured as follows. Section 2 presents our model to annotate service process models. Section 3 describes an overview of our approach, which is composed of (i) a filtering task, described in Section 4; (ii) a structural similarity evaluation task (Section 5); (iii) a preference satisfaction evaluation and ranking tasks (Section 6). Related work are discussed in Section 7. Finally, the conclusions are presented in Section 8.

2 BACKGROUND AND NOTATIONS

Our *S-Discovery* framework is based on matching techniques that operate on process models. The process model consists of a set of related activities that are organized using control flow structures to construct complex behavior models. To abstract as much as possible from any existing notation formalism, we represent a process model as a directed graph $p = (A, C, E, Q)$, called *process graph* (*p-graph* for short), where A is a set of activity nodes, C is a set of connector nodes, E is a set of directed edges and Q is a set of quality annotations. An activity node represents an atomic task which is described by its name (N), a set of inputs (In), a set of outputs (Out) and a set of quality annotations (Q). Notice that activity inputs and outputs are annotated unsing a domain ontology. Connector nodes describe control flows between activities, and represent Split and Join operators of types *XOR* or *AND*. Split connectors have multiple outgoing edges, while Join connectors have multiple incoming edges. Quality annotations are of the form (m, r), where r is a value for a QoS attribute m. They can characterize the process as a whole or specific activities.

A user query is a p-graph $q = (A, C, E, P)$, where A, C, E are as defined before and P is a set of QoS preferences, which are specified as expressions using the following constructors: *around*, *between*, *max*, *min*, *like* and *dislike*.

A preferred order between preferences can be defined using the complex constructors *pareto* ($\otimes$) and *prioritized* (&). The semantics of the terms of this vocabulary were taken from the *PreferenceSQL* approach (Kießling,), however the user may personalize these semantics by means of a membership function μ. User can label a preference as *hard* or *soft*, the difference being that a target p-graph must satisfy all hard preferences, while the satisfaction of soft preferences is optional.

Figures 1(a) and 1(b) show, respectively, a process model and a sample user query annotated with hard and soft preferences.

3 OVERVIEW OF THE S-DISCOVERY FRAMEWORK

Given an extensive repository of published p-graphs, the goal of our framework is to retrieve a ranked list of p-graphs that best fulfill a p-graph query. Our framework is a multi-stepped approach, as illustrated by Figure 2.

Given a query p-graph q, the *Filter* module selects the p-graphs that can most likely answer the query (p-graphs $T_1, \ldots, T_n$); it avoids scanning the whole repository to compare each target p-graph against the query. This module retrieves all the p-graphs sharing at least one activity with the query p-graph, and this is done by relying on an index built on an offline step.

This set is subsequently passed to the *Structural Similarity Evaluator* that measures the structural similarity $\lambda_{struc-i}$ between each selected p-graph (target) and the query. Furthermore, a set of activity mappings between query and target p-graphs is established (mappings M_i, $1 \leq i \leq n$).

Next, the *Preference Satisfaction Evaluator* computes the degree of satisfaction λ_{pref-i} of the QoS preferences at the basis of the mappings computed in the previous stage. At the end, the retrieved p-graphs are ranked according to their structural similarity and the preference satisfaction degrees using a set of aggregation metrics. The following sections present our contributions to each stage of the *S-Discovery* framework.

4 REPOSITORY FILTERING

As mentioned previously, a target p-graph is a potential match of a query p-graph when they have similar activities. To avoid comparing each query activity against all the activities stored in the repository, we defined an index structure over the activities of the repository p-graphs.

This index is built by assuming that activities having similar inputs/outputs are similar (Gater et al., 2011a). The structure we defined indexes the activities stored in the repository by attaching to each concept C of the ontology two sets C_{In} and C_{Out} recording the identifiers of the activities where it appears,

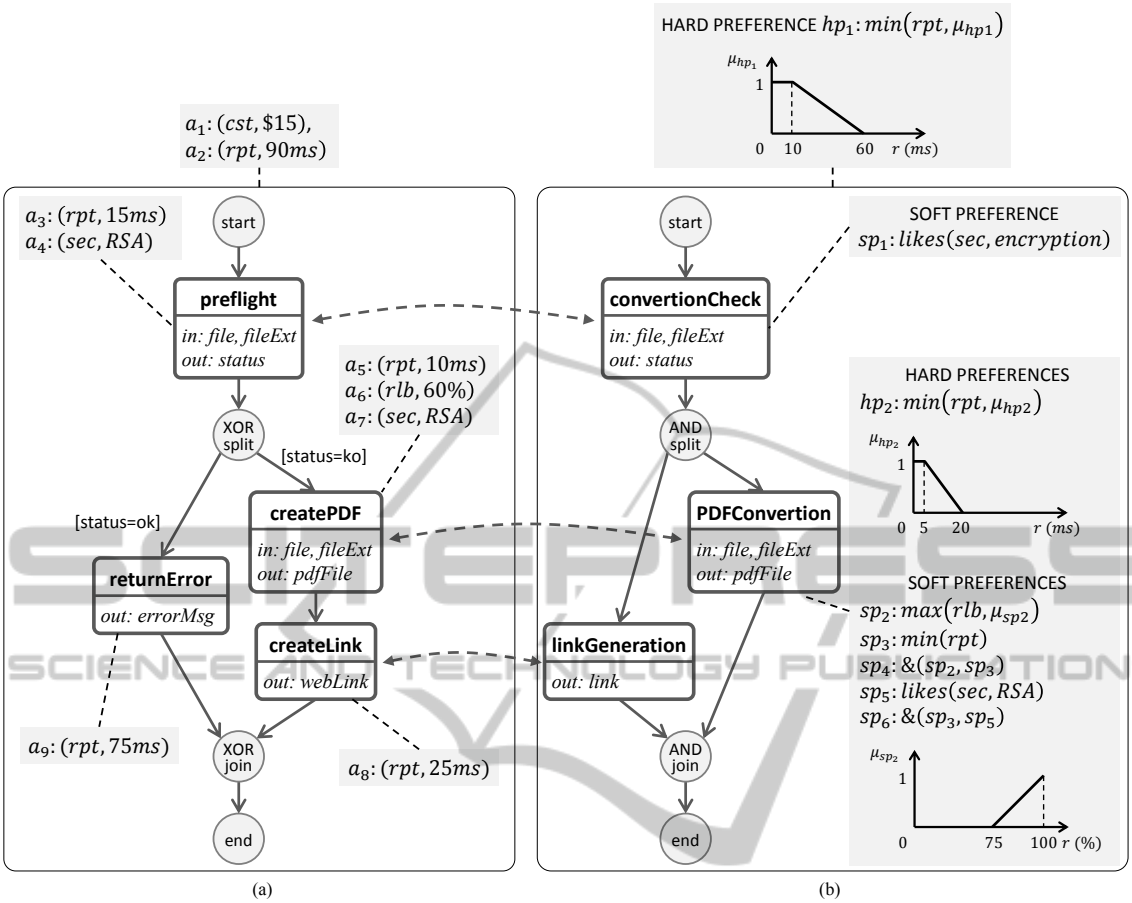

Figure 1: Mapping between a (a) sample target p-graph T_1 and a (b) sample query p-graph Q_1.

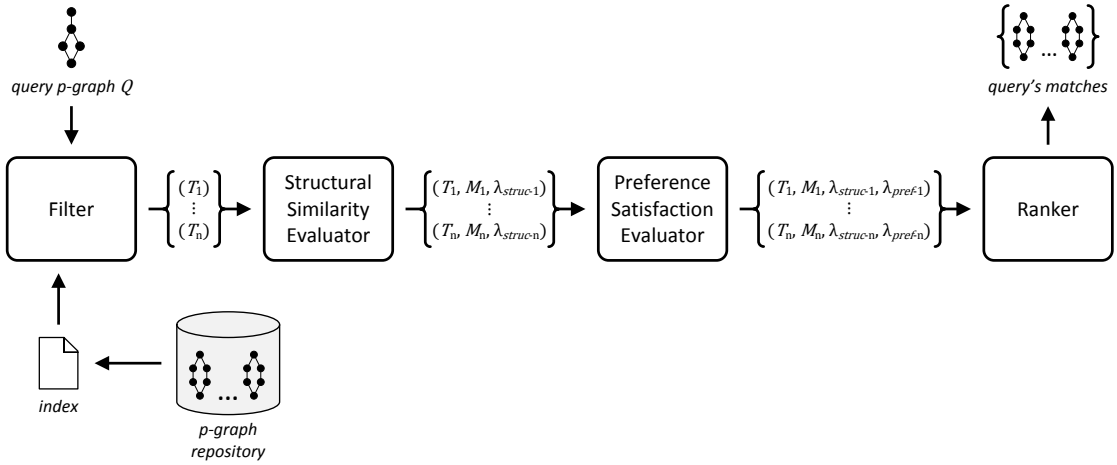

Figure 2: S-Discovery framework architecture.

respectively, as an input or an output. Furthermore, since we are not only interested in retrieving inexact matches, the technique should be able to retrieve the activities that don't match exactly the query activity but also those having similar inputs/outputs, i.e. the activities having inputs/outputs that are semantically related to those of the query activity. To avoid making this computation in real time, the set C_{In} (resp. C_{Out}) of a concept C record also the activities having as input (resp. output) a concept which is semantically related (its descendants, ascendant, ...) to C.

Thus given a query activity A_q, the set of its poten-

tial activity matches is composed of all the activities belonging to the set of annotations attached to the inputs and outputs of A_q. Straightforwardly, the set of process-match candidates of a query is the set of target p-graphs with whom it shares at least one activity.

5 STRUCTURAL SIMILARITY EVALUATION

The problem of process matching is reduced to a graph matching problem and an error-correcting subgraph isomorphism (ECSI for short) detection algorithm (Messmer, 1995) was adapted to this purpose. The principle behind ECSI algorithm is to apply edit operations (node/edge insertions, deletions and substitutions) over the target graph until there exists a subgraph isomorphism to the query graph. Each edit operation is assigned a cost function, on the basis of which the quality of an ECSI is estimated. The goal is then to find the sequence of edit operations leading to the ECSI between the query and target graphs that has the minimal cost. To find this sequence, the ECSI detection algorithm relies on an exhaustive A* search space algorithm.

The adaptation of this algorithm in order to handle the p-graphs concerns the definition of: (i) measures for evaluating the similarity of two activities that integrate the similarity of their names, inputs and outputs; (ii) measures for evaluating the behavioral/structural similarities; (iii) heuristics for detecting the granularity level differences.

This algorithm allows evaluating the similarity of two p-graphs as well as finding a set of correspondences between their activities. Experimental results showed the effectiveness of this approach in terms of precision/recall of the found matches. However, the time complexity induced by the combinatorial search space limits their application in practice to p-graphs of relatively small sizes (55 activities). To make this algorithm tractable, we defined two heuristics that aim to find matches having acceptable qualities in reasonable execution times. More details are given in (Gater et al., 2010; Gater et al., 2011b).

6 PREFERENCE SATISFACTION EVALUATION

After the calculation of the structural similarity between query and each candidate p-graph, the most similar ones are subjected to the preference satisfaction evaluation, which calculates the degree to which the QoS annotations of the p-graphs satisfy the QoS preferences of the query. The procedure first calculates the satisfaction degrees of atomic preferences and, then, it recalculates these degrees based on the complex preferences. At the end, a global satisfaction degree is obtained from the aggregation of the preference degrees. The mapping between query and target p-graphs is used in the evaluation procedure to recalculate the QoS attributes of target p-graphs and to evaluate the satisfaction degree of atomic preferences.

6.1 Atomic and Complex Preference Evaluation

For each activity correspondence (w,v), the degree to which each atomic preference of v is satisfied by its corresponding annotation in w is calculated. The same is similarly done for the profile preferences.

For a preference p of the type *around*, *between*, *min* or *max*, given its corresponding annotation a, the satisfaction degree $\delta(p,a)$ between them is given by the normalized *satisfaction distance* $d(p,a)$, which measures how far is the value r in annotation a from those favored by preference p. For a preference of the type *likes* or *dislikes*, the satisfaction degree is based on the semantic similarity between concepts given by the classic edge counting technique proposed in (Wu and Palmer, 1994). When a membership function μ defines the semantics of the preference, the satisfaction degree is a simple application of the function over the corresponding quality attribute.

When a hard preference is not satisfied, the target p-graph is eliminated from the discovery result. As a consequence, the verification of the hard preferences at activity level may be interwoven with the structural matching when the latter checks weather two activities match. This may eliminate the mappings expressing structural match but not expressing preference satisfaction and, thus, pruning the search space.

The satisfaction degrees of the atomic preferences are reevaluated according to the order of importance defined by the complex preferences (*pareto* and *prioritized*). The goal is to assign weights to the satisfaction degrees of atomic preferences to capture the order of importance defined by complex preferences. This is done with the help of a *preference graph*, which is a rooted directed graph whose nodes represent atomic preferences, edges represent a *prioritized* preference from source to destiny, and each node of the graph has weight $\omega_p = 1/i$, where i is the edge distance from the node to the graph root. Then, the reevaluation of a preference p is done by $\delta'(p,a) = \delta(p,a) \times \omega_p$.

6.2 Preference Satisfaction Degree Calculation

The *global preference satisfaction degree* λ_{pref} indicates the degree to which the QoS annotations of a target satisfy the QoS preferences of a query. This degree is calculated with the help of a *preference satisfaction metric*, which receives as input the preference satisfaction degrees $\{\delta_1, ..., \delta_k\}$ of the target and aggregates them to provide the global preference satisfaction degree λ_{pref}. Our framework provides three different metrics.

The *average-based metric* calculates the preference satisfaction degree λ_{pref} as the average of the preference satisfaction degrees $\{\delta_1, ..., \delta_k\}$.

The *linguistic quantifier-based metric* calculates the degree λ_{pref} by measuring the truth degree of the sentence γ: "almost all preferences are satisfied". This sentence is a fuzzy quantified proposition defined using a relative quantifier (e.g., *almost all, at least, around half*, etc.) (Glöckner, 2004).

The *bipolar-based metric* calculates the degree λ_{pref} by evaluating the bipolar condition (Dubois and Prade, 2008) "all hard preferences are satisfied and if possible at least one soft preference is satisfied". This method returns a bipolar degree of the form $\lambda_{pref} = (\delta_{hp}, \delta_{sp})$ meaning that "all hard preferences are satisfied to at least a degree of δ_{hp} and at least one soft preference is satisfied to at least a degree of δ_{sp}". More details are given in (Lemos et al., 2012).

Once the structural similarity and quality satisfaction degrees are computed, the retrieved p-graphs are subsequently ranked according to the structural and quality satisfaction degrees using aggregation techniques. Our framework proposes a set of aggregation techniques (lexicographic order, weighted average, fuzzy-based techniques) that are detailed in (Lemos et al., 2012).

7 RELATED WORK

Our work addresses an important topic in the area of service oriented architecture, which is the discovery of services based on their process models. Several work have been proposed similarity measures (Wombacher et al., ; de Medeiros et al., 2008; Dijkman et al.,) for the evaluation of the similarity of two service process models. These approaches proposed similarity measures that consider either the structural or behavioral perspectives of the process models. While structure-based approaches consider the process topologies, behavior-based approaches consider the execution semantics of the process models.

In this case, the service process discovery is done by comparing the query against each target service, and subsequently ranking target processes according to their closeness to the query. To avoid browsing the whole process repository, some approaches rely on indexing structures (Gater et al., 2011a; Awad and Sakr, 2010; Yan et al., 2010) to quickly retrieve the processes that are the most likely similar to a (part of) process query.

With regard to the quality-based service discovery, current approaches (Mokhtar et al., ; Agarwal et al., 2009) consider services as black boxes, so quality requirements are defined over the service profile. Generally, they specify quality preferences as relational expressions, fuzzy sets, linguistic variables, or utility functions. These approaches do not propose preference constructors to help user better define and compose his preferences and they are not abstract enough to be adapted to different non-functional contexts.

While process similarity search is an active field in the domain of business process management research area, little attention was given until now to the discovery of the services hosted in the cloud (Goscinski and Brock, 2010); the existing techniques are limited in what information can be used when publishing and discovering services (Microsoft Azure (Microsoft, Inc.,)). To the best of our knowledge there are no process discovery framework allowing to combine functional and non-functional requirements.

8 CONCLUSIONS

In this paper we presented a framework for process discovery taking into account both functional and non-functional criteria. User query is expressed as a process model adorned with quality annotations expressing user preferences and requirements. Our approach will allow searching process repositories offered by BPaaS providers.

In our past work, we implemented basic matching operators and evaluated them in terms of efficiency and effectiveness. Our future work consists in building a prototype implementing the framework presented in this paper by reusing and adapting our matching operators.

REFERENCES

Agarwal, S., Lamparter, S., and Studer, R. (2009). Making Web services tradable: A policy-based approach for specifying preferences on Web service properties. *Journal of Web Semantics*, 7(1):11–20.

Anstett, T., Leymann, F., Mietzner, R., and Strauch, S. (2009). Towards bpel in the cloud: Exploiting different delivery models for the execution of business processes. In *SERVICES'2009)*, pages 670–677.

Awad, A. and Sakr, S. (2010). Querying graph-based repositories of business process models. In *DASFAA Workshops'10*.

de Medeiros, A. K. A., van der Aalst, W. M. P., and Weijters, A. (2008). Quantifying process equivalence based on observed behavior. In *Journal DKE*, pages 55–74.

Dijkman, R., Dumas, M., and García-Bañuelos, L. Graph matching algorithms for business process model similarity search. In *BPM'09)*.

Dubois, D. and Prade, H. (2008). Handling bipolar queries in fuzzy information processing. In *Handbook of Research on Fuzzy Information Processing in Databases*, pages 97–114. IGI Global.

Gater, A., Grigori, D., and Bouzeghoub, M. (2010). Complex mapping discovery for semantic process model alignment. In *IIWAS'10*, pages 317–324.

Gater, A., Grigori, D., and Bouzeghoub, M. (2011a). Indexing process model flow dependencies for similarity search. In *COOPIS'2012*.

Gater, A., Grigori, D., Haddad, M., Bouzeghoub, M., and Kheddouci, H. (2011b). A summary-based approach for enhancing process model matchmaking. In *SOCA'11*, pages 1–8.

Glöckner, I. (2004). *Fuzzy Quantifiers in Natural Language: Semantics and Computational Models*. Der Andere Verlag.

Goderis, A., Li, P., and Goble, C. A. (2006). Workflow discovery: the problem, a case study from e-Science and a graph-based solution. In *ICWS'06*, pages 312–319.

Goscinski, A. and Brock, M. (2010). Toward dynamic and attribute based publication, discovery and selection for cloud computing. *FGCS'10*.

Kießling, W. Foundations of preferences in database systems. In *VLDB'02*.

Lemos, F., Abbaci, K., Grigori, D., Hadjali, A., Bouzeghoub, M., Liétard, L., and Rocacher, D. (2012). Intégration de préférences dans la découverte et la sélection des services web: Une approche fondée sur les ensembles flous. *Ingénierie des Systèmes d'Information*.

Messmer, B. T. (1995). *Graph Matching Algorithms and Applications*. PhD thesis, University of Bern, Switzerland.

Microsoft, Inc. Microsoft Azure. http://www.windowsazure.com.

Mokhtar, S. B., Preuveneers, D., Georgantas, N., Issarny, V., and Berbers, Y. EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. *JSS'08*.

Pathirage, M., Perera, S., Kumara, I., and Weerawarana, S. (2011). A multi-tenant architecture for business process executions. In *ICWS'11*, pages 121–128.

Wombacher, A., Fankhauser, P., Mahleko, B., and Neuhold, E. Matchmaking for business processes based on choreographies. In *EEE'04*.

Wu, Z. and Palmer, M. S. (1994). Verb semantics and lexical selection. In *32nd Annual Meeting of the Association for Computational Linguistics (ACL 1994)*, pages 133–138.

Yan, Z., Dijkman, R. M., and Grefen, P. (2010). Fast business process similarity search with feature-based similarity estimation. In *CoopIS'10*.