Adding Non-functional Preferences to Service Discovery

Fernando Lemos[1], Daniela Grigori[2], and Mokrane Bouzeghoub[1]

[1] Versailles University, 45 Av. des États Unis 78000 Versailles, France
{fernando.lemos,mokrane.bouzeghoub}@prism.uvsq.fr
[2] Paris-Dauphine Univ., Pl. Maréchal de Lattre de Tassigny 75775 Paris, France
daniela.grigori@dauphine.fr

Abstract. The growth of the number of published services rendered searching for a specific service within repositories a critical issue. In this paper, we present an approach to extend structure-based service discovery by making it sensitive to user preferences over service quality defined at different granularity levels of the service structure.

Keywords: Web services, QoS, preferences, process model matching.

1 Introduction

In the last years, the number of published services has been increasingly growing since more and more organizations invested on service management practices. However, this growth rendered searching for a specific service within repositories a critical issue for the success of service computing in general. For the functional aspect of the search, some approaches allow users to detail the process model (PM) describing the structure of the requested service, and thus PM matching techniques have been proposed to find the services best matching the query. However, current PM matching approaches [1,2] still return a large number of services offering similar functionalities [2]. On the non-functional aspect of the search, non-functional requirements such as quality preferences (e.g., response time) are one way to discriminate between structurally similar services. Nevertheless, current works consider Web services as black boxes, limiting the approaches to the profile level [3,4,5], which is not sufficient and do not fulfill user needs as non-functional aspects can be *hidden within the specification of the service structure*.

In our vision, service discovery should be based on both structural specification and non-functional aspects of services. Targeting this goal poses challenges at two levels. (i) At the description level, provide a formal model that allows one to specify, at different granularity levels, non-functional attributes as annotations of the functional specification; and allow the user to enrich his query with (required and preferred) non-functional requirements. (ii) At the discovery level, define a similarity measure aggregating both functional and non-functional similarities and provide algorithms combining the structural matching and the non-functional matching.

M. Brambilla, T. Tokuda, and R. Tolksdorf (Eds.): ICWE 2012, LNCS 7387, pp. 299–306, 2012.

In this work, we extend service matching algorithms based on the PM specification by making them sensitive to user preferences concerning service quality. Our contributions to the above challenges are: (i) we extend the PM representing service structure with adornments for non-functional factors. Each annotation is defined either at the activity level or at level of the service itself. The user query is also a PM complemented with a set of selection clauses, which are defined either as required or preferred criteria in order to avoid empty or overloading answers. (ii) The service discovery is seen as a matching process between the user query PM and a target PM, in which, at the different stages, quality preferences are taken into account. To the best of our knowledge, there is no other approach addressing user preferences on quality factors in the service matching process.

Section 2 presents our model. Section 3 details our approach and experimental results. Section 4 discusses related works. Section 5 concludes the paper.

2 Abstract Representation of Service Process Model

Process models consists of a set of atomic activities, combined using control flow structures to construct complex processes. To abstract from a specific PM description language (e.g., WS-BPEL and OWL-S) and provide a broader general approach, we introduce a graph-based model, as follows. A **process model** is a directed labeled graph $G = (V, E)$, where V is a set of activities and connector nodes and E is a set of edges. An **activity node** is described by its name, inputs and outputs. **Connector nodes** are: *(i) start* or *end*, representing the beginning or the termination of the process execution, respectively; (ii) *AND-split*, triggering all of its outgoing concurrent branches that are synchronized by a corresponding (iii) *AND-join*; (iv) *XOR-split,* representing a choice between one of several alternative branches that are merged by a corresponding (v) *XOR-join.*

For example, the service PM graph depicted in Figure 1(a) converts common types of documents to PDF. It receives as input a file and its extension and executes a pre-flight activity to check whether the file can be converted. If so, *createPDF* activity converts the file to PDF and activity *createLink* returns a link so user can download the converted file. Otherwise, an error message is sent.

QoS information is added by service providers as graph **annotations** of the form (m, r), where r is a value for a QoS attribute m. They can characterize the service as a whole (**profile annotations**) or specific activities (**activity annotations**). Figure 1(a) shows the previous service PM graph adorned with the profile annotations a_1 and a_2 indicating the cost and response time and several activity annotations a_3 to a_9 indicating the response time, reliability and security. We precise that service PMs are considered to be already annotated with QoS attributes by their providers using techniques like in [6].

Providers can also define aggregation functions to automatically calculate global QoS information from activity annotations. An **aggregation function** is a function of the form $f_{[m]} : G \rightarrow R$, where m is a QoS attribute, G is a PM graph and R is a set of atomic values. We denote by F the **set of aggregation functions**. Specifications of such functions can be found in [6].

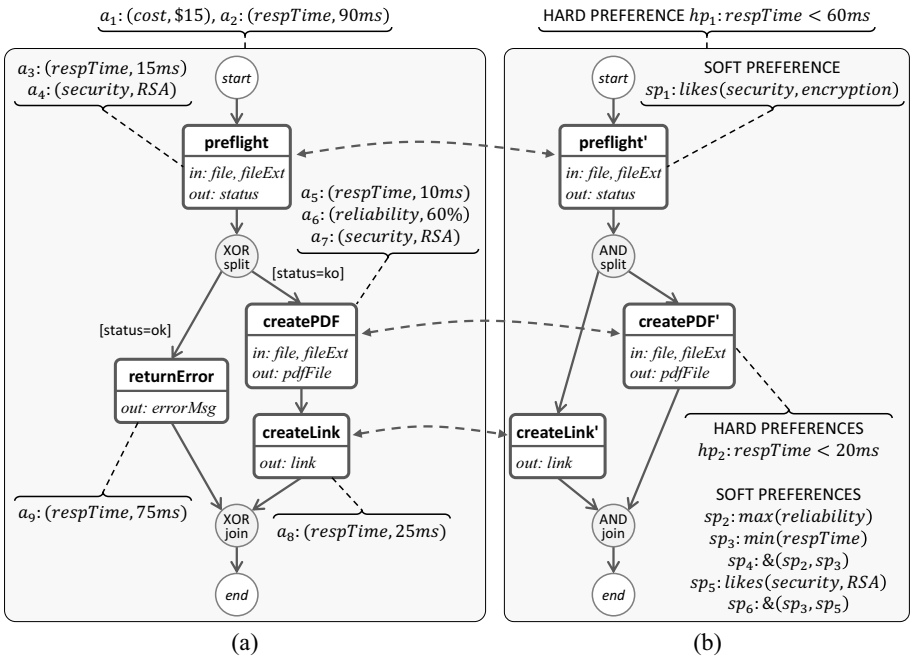

Fig. 1. Mapping between (a) target graph T_1 and (b) query graph Q_1

A user query is specified by (i) a PM graph describing structural requirements and (ii) a set of preferences describing QoS requirements, which are defined for the service as a whole (**profile preferences**) or for specific activities (**activity preferences**). Preferences can also be: (i) **hard**, when they must be satisfied and they are specified as relational expressions of the form (m, o, r), where o is a relational operator and r is a value for QoS attribute m[1]; or (ii) **soft**, when their satisfaction is optional, but desirable.

A soft preference is specified using a subset of the preference constructors proposed by *Preference SQL* [7], one of the first ones to provide a declarative and semantically intuitive model of preferences. The constructors are: (i) $around\,(m, r_{desired})$: it favors the value $r_{desired}$ for attribute m; otherwise, it favors those close to $r_{desired}$; (ii) $between\,(m, r_{low}, r_{up})$: it favors the values inside the interval $[r_{low}, r_{up}]$; otherwise, it favors those close to the limits; (iii) $max\,(m)$: it favors the highest value; otherwise, the closest value to the maximum is favored; (iv) $min\,(m)$: it favors the lowest value; otherwise, the closest value to the minimum is favored; (v) $likes\,(m, r_{desired})$: it favors the value $r_{desired}$; otherwise, any other value is accepted; (vi) $dislikes\,(m, r_{undesired})$: it favors the values different from $r_{undesired}$; otherwise, $r_{undesired}$ is accepted; (vii) $\otimes\,(p_i, p_j)$: it states that the soft preferences p_i and p_j are equally important; (viii) $\&\,(p_i, p_j)$: it states that the soft preference p_i is more important than the soft preference p_j.

[1] *We abstract from the different units in which a value can be described.*

Preference SQL distinguishes two types of preferences: *atomic* (*around*, *between*, *max*, *min*, *likes* and *dislikes*) and *complex* ($\otimes$ and &). It also distinguishes two types of atomic preferences: *numerical* (*around*, *between*, *max* and *min*) and *non-numerical* (*likes* and *dislikes*). In this work, the values in non-numerical preferences are taken from a global ontology O given by the user. As specified, complex preferences can be defined over existent complex preferences.

Figure 1(b) shows a sample user query annotated with hard and soft preferences: (i) the profile preference hp_1 indicates the response time must be less than 60 ms; (ii) the soft preference sp_2 indicates that user prefers services having activity B with maximal reliability; (iii) the complex preference sp_4 indicates that to satisfy preference sp_2 is more important than the satisfaction of sp_3.

3 Dealing with Preferences in Service Discovery

The evaluation of query preferences is strongly dependent of a structural mapping between the PMs of query and target services, as described in Subsections 3.1 and 3.2. An important class of solutions to the problem of finding a mapping between PMs is that of *approximate matching* algorithms [1,2], that allow to find target PMs similar to user query. Early approaches of this class reduce the problem to the discovery of a (sub) graph isomorphism between two PMs [2].

Our recent work [2] proposes an algorithm based on state-space searching to discover the best mapping between two PM graphs. To reduce the space search, a *pruning function* is proposed. The returned mapping has a *structural similarity* SS that defines a total order between targets [2], but cannot distinguish between graphs having similar structure and different quality. Moreover, targets very similar to the query and better satisfying the preferences should top the ranking. For these reasons, we extend the PM matching by: (i) evaluating hard preferences during the matching task to reduce the space-search; and (ii) evaluating the soft preferences to rank potential graphs considering structural and quality aspects.

3.1 Evaluating Hard Preferences in Service Matching

The evaluation of query profile preferences against target profile annotations may reduce the number of target service PMs to be matched. However, the structural mapping between query and target may "change" some profile attributes. For example, by considering the matching between Q_1 and T_1 in Figure 1, found by a matching algorithm like in [2], the trace containing activity *returnError* will never be consumed (executed). Thus, recalculating the response time of T_1 ignoring activity *returnError* gives 50 ms. According to the profile preference hp_1, if the recalculation had not been done, T_1 would be discarded.

The recalculation of profile annotations is done over the target's *consumable graph*, which is a graph containing only the consumable paths of the target according to its structural mapping with the query. More formally, a **consumable graph** of a graph G w.r.t. to a mapping M is the graph obtained by eliminating from each block b of G the branches containing no activity mapped by M.

A **block** is any subgraph limited to a split node, its respective join node and the branches between them. Therefore, our algorithm is composed of two steps:

Step 1: Evaluation of Hard Activity Preferences. The first step in evaluating the hard preferences of a query activity is to discover the target activity that semantically corresponds to it. For this, we propose to extend the pruning technique of the PM matching algorithm described in [2] to also discard non-promising mappings according to hard preferences. Thus, a target activity semantically equivalent to a query activity must also satisfy all the hard preferences of the query activity. Given an activity hard preference $hp = (m, o, r)$ and a target annotation $a = (m, v)$, a satisfies hp iff the expression (v, o, r) is true.

Step 2: Evaluation of Hard Profile Preferences. Once a target satisfying all activity preferences is discovered, its hard profile preferences are evaluated. The evaluation algorithm (i) recalculates the profile annotations using the consumable graph and the aggregation functions, and then (ii) checks if all hard profile preferences are satisfied by the target profile annotations. In Figure 1, the consumable graph of T_1 satisfies all the hard preferences of Q_1.

3.2 Dealing with Soft Preferences in Service Selection

The *satisfaction degree* (δ) is our metric to define how well the annotations of a target satisfy the soft preferences of a user query. First, we calculate the satisfaction degree between each soft atomic preference and its corresponding annotation. Then, we aggregate the satisfaction degrees of atomic preferences according to the order of importance defined by the complex preferences.

The **Evaluation of Soft Atomic Preferences** depends on their type. For a numerical preference p, given its corresponding annotation $a = (m, r)$, the satisfaction degree $\delta(p, a)$ between them is given by the equation $\delta(p, a) = {}^{1}/_{(1+d(p,a))}$. This equation normalizes the *Satisfaction Distance* $d(p, a)$, which measures how far is the value r in annotation a from those favored by preference p. The satisfaction distance depends on the type of p as described in Table 1.

For non-numerical preferences, the satisfaction degree is based on the semantic similarity between concepts given by $wp(O_G, c_1, c_2)$, where c_1 and c_2 are the concepts to be compared according to an ontology O_G. Among the similarity metrics defined in the literature [8], we applied the classic edge counting technique proposed in [9]. Given a non-numerical preference p and an annotation a, the satisfaction degree $\delta(p, a)$ between them is presented in Table 2.

Based on the mapping of Figure 1 and on the ontology in Figure 2, the satisfaction degrees of soft preferences of query Q_1 are $\delta(sp_1, a_4) = 1$, $\delta(sp_2, a_6) = 0.03$, $\delta(sp_3, a_5) = 0.09$ and $\delta(sp_5, a_7) = 1$, where $d(sp_2, a_6) = 40$ and $d(sp_3, a_5) = 10$.

The **Evaluation of Soft Complex Preferences** aims, at first, to assign weights to the satisfaction degrees of atomic preferences to capture the order of importance defined by complex preferences. Then, these weighted degrees are aggregated to provide the satisfaction degree between the query and the target. The evaluation the complex preferences is composed of the following steps:

Table 1. Satisfaction distance of numerical preference p w.r.t. annotation $a = (m, r)$

Numerical Preference p	*Satisfaction Distance* $d(p, a)$
$around\,(m, r_{desired})$	$d\,(p, a) = \lvert r - r_{desired}\rvert$
$between\,(m, r_{low}, r_{up})$	$d\,(p, a) = \begin{cases} 0, & r \in [low, up] \\ low - r, & r < low \\ r - up, & r > up \end{cases}$
$max\,(m)$	$d\,(p, a) = r_{max} - r$, where r_{max} is the highest value
$min\,(m)$	$d\,(p, a) = r - r_{min}$, where r_{min} is the lowest value

Table 2. Satisfaction degree of non-numerical preference p w.r.t. annotation $a = (m, r)$

Non-numerical Preference p	*Satisfaction Degree* $\delta(p, a)$
$likes\,(m, r_{desired})$	$\delta\,(p, a) = \begin{cases} 1, & r_{desired} = r \\ 1, & r_{desired} \text{ subsumes } r \\ wp(O_G, r_{desired}, r), & otherwise \end{cases}$
$dislikes\,(m, r_{undesired})$	$\delta\,(p, a) = 1 - likes\,(m, r_{undesired})$

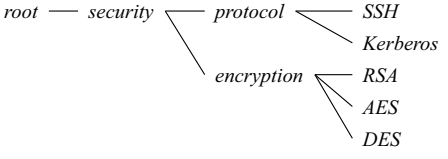

Fig. 2. Sample Security ontology

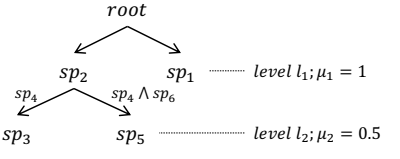

Fig. 3. Preference tree of query Q_1

Step 1. We construct a *preference tree* t_{sp} whose nodes represent atomic preferences, edges represent a *prioritized* (&) preference, from parent to child, and each level l_i of the tree has weight $\mu_i = 1/i$. We denote by $p.l$ the level assigned to preference p. We consider that user has not defined any contradictory preference.

The construction of the tree first addresses each preference $\&\,(p_i, p_j)$ by (i) if $p_i.l = null$, then $p_i.l \leftarrow l_1$ and $p_j.l \leftarrow l_2$, and (ii) if $p_i.l \neq null$, then $p_j \leftarrow p_i.l+1$. Next, it evaluates each preference $\otimes\,(p_i, p_j)$ by applying the following rules:

Rule 1: $p_i.l \neq null \wedge p_j.l = null$ then $p_j.l \leftarrow p_i.l$; *Rule 2:* $p_i.l = null \wedge p_j.l \neq null$ then $p_i.l \leftarrow p_j.l$; *Rule 3:* $p_i.l = null \wedge p_j.l = null$ then $p_i.l \leftarrow l_1$ and $p_j.l \leftarrow l_1$; *Rule 4:* $p_i.l \neq null \wedge p_j.l \neq null$ then: (a) $p_i.l < p_j.l$ then $p_j.l \leftarrow p_i.l$ and the levels of p_j descendants are updated accordingly; (b) $p_j.l < p_i.l$ then $p_i.l \leftarrow p_j.l$ and the levels of p_j descendants are updated accordingly; *Rule 5:* level l_1 is assigned to remaining preferences. Figure 3 shows the tree of Q_1.

Step 2. The satisfaction degree between a query Q and a target T w.r.t. a mapping M is given by $\delta\,(Q, T, M) = \sum_{p \in S_{sp}} \delta(p,a) \times \mu_{p.l} / \sum_{p \in S_{sp}} \mu_{p.l}$, where a is the annotation corresponding to the QoS attribute of preference p. This equation is a sum of the satisfaction degrees of atomic preferences affected by the weights of their levels in the tree. In our example, $\delta\,(Q_1, T_1, M_1) = 0.53$.

3.3 Service Ranking Based on Structural and Quality Aspects

Two classic methods are used to order the potential targets of a given query according to structural and quality aspects. The first is the *lexicographic order*: targets are ordered according to the structural similarity degree SS and the preference satisfaction degree is used to break ties. The second is the *weighted average* $wa\,(TQ, M) = \mu_{SS} \times SS\,(M) + (1 - \mu_{SS}) \times \delta\,(Q, T, M)$, where $0 < \mu_{SS} < 1$ is the weight assigned to the semantic similarity degree. The user can specify the contribution of each degree to the calculation of the overall similarity.

3.4 Preliminary Experimental Results

To evaluate our approach, we implemented a prototype on top of the service matching platform proposed by [2]. Our experiments considered 64 services of average size of 15 activities and providing 12 quality properties. The first experiments measured the evaluation time of (i) hard preferences in the matching algorithm and (ii) soft preferences after the matching step. In both cases, the extra time represents less than 1% of the matching time.

The last experiments measured the ranking effectiveness. Clearly, a discovery process that takes into account the quality aspect beyond the structural one provides better responses than a structure-based method. Thus, we were interested in measuring how close is the ranking of our solution compared to the ranking of an expert. For this, an expert manually compared each query to each target and noted it in a *Likert* scale. Then, the results were sorted according to their similarities and compared with our ranking using the NDCG formula. The results obtained for weighted average and lexicographic order rankings were 0.996967 and 0.998752, respectively, which shows that our solution provides a ranking that is strongly close to that defined by the experts in all of our experiments.

4 Related Work

Many approaches for service retrieval based on non-functional characteristics have been proposed in the literature [3,10,11,4]. In these works, quality preferences are specified by (i) relational expressions [3], evaluated to a distance between the preference and the QoS information provided by the service; (ii) fuzzy sets [10], described by membership functions mapping each value of quality attributes to the degree at which the user is satisfied with it; (iii) linguistic variables [4], whose values are terms (e.g., *fast*, *slow*) and whose evaluation returns a match degree in a qualitative scale; or (iv) utility functions [11], similar to fuzzy sets, but can be specified over a discontinuous domain.

The order of importance between preferences is not addressed by these approaches. Instead weights are attributed to QoS properties to be multiplied with the satisfaction degrees of the preferences. These weights are specified by the user at query definition time [11], by an expert at design time [10], or they are fixed in the evaluation process [3]. The aggregation of satisfaction degrees is

done via aggregation functions like the sum [3,11], via solutions to the *constraint satisfaction problem* [10], or using match degrees in a qualitative scale [4].

These approaches do not propose preference constructors to help user better define his preferences and they are not abstract enough to be adapted to different non-functional contexts. More important, these approaches consider services as black boxes, so quality requirements for internal activities are not addressed.

5 Conclusions

We presented an approach for service discovery considering structure and quality requirements. First, we proposed a formal model to annotate service PMs with quality properties and user queries with quality preferences. Then, we showed how preferences are addressed in the service discovery process. Our approach can be easily applied to other non-functional requirements. As future work, we intend to study preferences considering user's viewpoint and semantic compositions of structural similarity and preference satisfaction.

Acknowledgment. This work has received support from the French National Agency for Research (ANR) on the reference ANR-08-CORD-009.

References

1. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
2. Grigori, D., Corrales, J.C., Bouzeghoub, M., Gater, A.: Ranking bpel processes for service discovery. IEEE TSC 3, 178–192 (2010)
3. Mokhtar, S.B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y.: Easy: Efficient semantic service discovery in pervasive computing environments with QoS and context support. Journal of Systems and Software 81(5), 785–808 (2008)
4. Şora, I., Lazăr, G., Lung, S.: Mapping a fuzzy logic approach for QoS-aware service selection on current web service standards. In: ICCC-CONTI, pp. 553–558 (2010)
5. Zhang, Y., Huang, H., Yang, D., Zhang, H., Chao, H.C., Huang, Y.M.: Bring QoS to P2P-based semantic service discovery for the universal network. Personal Ubiquitous Computing 13(7), 471–477 (2009)
6. Dumas, M., García-Bañuelos, L., Polyvyanyy, A., Yang, Y., Zhang, L.: Aggregate Quality of Service Computation for Composite Services. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 213–227. Springer, Heidelberg (2010)
7. Kießling, W.: Foundations of preferences in database systems. In: VLDB, pp. 311–322 (2002)
8. Cross, V.: Fuzzy semantic distance measures between ontological concepts. In: NAFIPS, vol. 2, pp. 635–640 (2004)
9. Wu, Z., Palmer, M.S.: Verb semantics and lexical selection. In: ACL, pp. 133–138 (1994)
10. Xiong, P., Fan, Y.: QoS-aware web service selection by a synthetic weight. In: FSKD, pp. 632–637 (2007)
11. Agarwal, S., Lamparter, S., Studer, R.: Making web services tradable: A policy-based approach for specifying preferences on web service properties. JWS 7(1), 11–20 (2009)