

# An Extensible and Personalized Approach to QoS-enabled Service Discovery\*

Le-Hung Vu, Fabio Porto, Karl Aberer  
Swiss Federal Institute of Technology  
CH-1015 Lausanne, Switzerland  
{lehung.vu, fabio.porto, karl.aberer}@epfl.ch

Manfred Hauswirth  
Digital Enterprise Research Institute  
National University of Ireland  
manfred.hauswirth@deri.org

## Abstract

*We present an extensible and customizable framework for the autonomous discovery of Semantic Web services based on their QoS properties. Using semantic technologies, users can specify the QoS matching model and customize the ranking of services flexibly according to their preferences. The formal modeling of the discovery process as an adaptive query execution plan facilitates the introduction of different discovery algorithms and the automatic generation of parallelized matchmaking evaluations. This enables adapting our approach to unpredictable arrival rates of user queries and scales up to high numbers of published service descriptions.*

## 1 Introduction

In the context of autonomous service usage, users have to be able to discover those services fulfilling their requirements for functional and non-functional properties. QoS is often the decisive criterion for a user to select a specific service among several functionally equivalent ones and in many cases, is the key of a provider's business success. For example, comparing two hotel reservation services from two different hotel Web sites offering similar capabilities in terms of online booking service, a user would lean towards the one with the shorter reservation response time, offering better price conditions, more comfortable rooms and higher customer-care facilities. Many types of services have QoS features as their main differentiating criteria, for example, file sharing/hosting, Internet TV/radio stations, music

stores, or teleconferencing services.

Different from the discovery of services matching functional requirements, the discovery of services based on their QoS is more complicated and the *personalization* of the QoS-based discovery process to adapt to different user's needs is an important requirement impacting on a number of issues: firstly, a Web service description can be used as an electronic advertisement of a real-life service that includes many domain-dependent QoS properties. Such properties are dynamic and depend on many factors, mostly the related user-side contextual or environmental conditions. Secondly, the advertisement of quality in a service description should only be considered as a claim, which the provider offers under certain conditions and to be verified and validated over time. This perception of the reputation information of the services is a subjective process by itself. Thirdly, the suitability of the service to a requirement of the user in terms of a QoS criterion is subject to her own needs. For example, the conclusion whether a certain deviation of quality is still acceptable should only be determined by the user. Finally, users want to obtain the most relevant matched services that are ranked according to their personalized preferences, especially because each of the services can provide many quality properties at different levels and with various reputation scores.

In this paper, we present our solution for the ontology-based discovery of Semantic Web services w.r.t. their QoS properties. Our goal is to automatically find those service descriptions that match the requirements of the users both in terms of functionalities and QoS, assuming that the actual negotiation, selection, and execution of the service can be done later by the user given the result of the service discovery. Extending our previous work in [1, 2], in this paper we introduce a complete discovery solution which combines our previously developed techniques and exploits semantic technologies to enable the personalization of the whole QoS-enabled service discovery process. Specifically, our approach has the following advantages:

- **Expressive and extensible conceptual modeling of service QoS:** Given the above complexity and dy-

\*The work presented in this paper was (partly) carried out in the framework of the EPFL Center for Global Computing and was supported by the Swiss National Funding Agency OFES as part of the European project DIP (Data, Information, and Process Integration with Semantic Web Services) No 507483 and by the Swiss National Science Foundation as part of the project: Computational Reputation Mechanisms for Enabling Peer-to-Peer Commerce in Decentralized Networks Contract No. 205121-105287. Manfred Hauswirth was supported by the Lón project funded by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

dynamic of QoS information, we propose an adequate semantic conceptual modeling approach for the flexible specification of user requirements and the QoS offerings of service, which is simple yet expressive. We can easily implement this conceptual model so that it is compatible with most of the current standards and approaches [3–7].

- **Customizable matchmaking model:** By exploiting semantic technologies, especially rule-based languages and reasoners, users and providers can express their own matching algorithms and preferences flexibly. The QoS-enabled discovery process can be done autonomously by reasoning on the constructed knowledge-bases based on these personalized settings.
- **Personalized ranking algorithm:** We provide useful and informative ranking results supporting a user in the selection of the most appropriate services, taking into account different quality dimensions of the services, their reputation, as well as preferences of the user.
- **Flexible and scalable implementation:** The discovery engine is modeled as an *adaptive query processing* system in which the basic steps of filtering, matchmaking, reputation-based QoS assessment, and ranking of services correspond to logical algebraic operators. This formal modeling enables us to apply cost-based optimization strategies to parallelize the evaluation of the expensive operators, considering that there can be an unpredictable number of queries from many users and that the number of published Web service descriptions may increase substantially in the future. Moreover, it facilitates the *plugging-in, testing, and comparison* of different algorithms on-the-fly.
- **Implemented prototype:** Our prototype validates our approach and confirms the usefulness of semantic technologies in dealing with the above issues. Though we adopt the WSMO ontology framework in our implementation as a proof-of-concept of our work, our approach is generally applicable to other models, e.g., to OWL-S+SWRL [8]. Our prototype as well as the online demonstration, related ontologies, and documentation are freely available<sup>1</sup> and demonstrate how Semantic Web technologies can be exploited in real-world applications.

In the next section we will present our conceptual QoS model. Our personalized matching and ranking algorithms are described in detail in Sections 3 and 4. Section 5 presents the formal discovery process model, followed by a brief description of the discovery prototype and analytical and experimental results in Section 6. We review the related work in Section 7 before ending with our conclusions.

<sup>1</sup><http://sirpeople.epfl.ch/thvu/download/qosdisc/>

## 2 Semantic Modeling of QoS

### Conceptual Modeling

Generally, semantically-annotated Web services are described by conjunctive sets of properties  $F \wedge Q$ , where  $F$  is the functional description and  $Q$  defines the QoS offerings. In this work, we focus on the modeling of a service's QoS and reuse the functionality description as specified in the WSMO framework [9]. Our conceptual model is developed mostly for the discovery of services based on their QoS properties and serves as a complement to the WSMO conceptual model. The negotiation between service providers and a user is considered as a later step after the user has already obtained the result from the service discovery and thus is out of the scope of this paper.

We describe a QoS offering  $Q$  in the service description as a set  $\{\langle C'_1(qi_1), cnd_1, P_1 \rangle, \dots, \langle C'_n(qi_n), cnd_n, P_n \rangle\}$ , where  $C'_k(qi_k)$ ,  $1 \leq k \leq n$  is the concept expression that constrains the instance  $qi_k$  of a QoS concept  $q_k$  in a QoS domain ontology.  $cnd_k$  is an axiom describing the environment (context) in which the provider commits to offer  $C'_k$ , and  $P_k$  is the set of preference rules of the provider. We also refer to  $cnd_k$  as the *context* to achieve the *QoS level*  $C'_k(qi_k)$ . For example, an online file hosting service specifies  $C'_k(qi_k) = \{\text{uploadSpeed} \geq 100\text{KBps}\}$  as the average upload speed that it offers, and  $cnd_k = \{\text{internetSpeed} \geq 1\text{Mbps}, \text{noFilesUploading} = 1, \text{price} = 10\$\}$  is the contextual conditions required by the provider to get the specified average upload speed. Note that in the examples to follow we simply differentiate an ontological concept UploadSpeed from its corresponding instance uploadSpeed by the capitalization of the first letter. The preferences  $P_k$  of the provider are a set of rules associating each logical expression in  $cnd_k$  with a set of matching results. For instance, the following rules in  $P_k$  describe how well a requester satisfies the price demanded by the provider.

$$\begin{aligned} \text{userPrice} \in \text{Price} \wedge \text{userPrice} \geq \text{price} &\rightarrow \text{prioritizedClient} \\ \text{userPrice} \in \text{Price} \wedge \text{userPrice} < \text{price} &\rightarrow \text{acceptedClient} \end{aligned}$$

Such preference above is currently used by a provider to specify that the requirement on an environmental condition, e.g.,  $\text{price} = 10\%$ , is optional. However, it can also be useful for a provider in deciding whether to offer service to a certain user in later negotiation steps.

Similarly, a user query (or a user goal) consists of the description of the functionality and the QoS a Web service should offer to fulfill a user's needs. A QoS requirement in user queries is symmetric to its counterpart in Web service descriptions. Web service consumers indicate by  $C'_k(qi_k)$  the condition on QoS parameter instances  $qi_k$  they are willing to accept, e.g.,  $C'_k = \{\text{reqUploadSpeed} \geq 20\text{KBps}\}$ . This QoS requirement is complemented by the contextual conditions  $cnd_k$  the client is able to agree with, e.g.,  $cnd_k =$

{userInternetSpeed = adsl1Mbps, noFilesUploading = 1, userPrice = 15\$}. The set of preferences rules  $P_k$  in the goal model comprises various settings for the discovery process: the matching levels for the QoS required by the user, which quality parameters are preferable by the users, how much the user trusts the reputation information on a service, etc. For example, a user wants to state that the requirement on the quality parameter ReqUploadSpeed, is optional. In addition, she wants the discovery component to automatically classify how well a service (with a quality parameter qs) satisfies her requirement. The following rules describe these preferences:

$$\begin{aligned} qs \in \text{ReqUploadSpeed} \wedge qs \geq 100\text{KBps} &\rightarrow \text{excellent} \\ qs \in \text{ReqUploadSpeed} \wedge \text{reqUploadSpeed} \leq qs < 100\text{KBps} &\rightarrow \text{good} \\ qs \in \text{ReqUploadSpeed} \wedge qs < \text{reqUploadSpeed} &\rightarrow \text{acceptable} \\ \forall qs \neg (qs \in \text{ReqUploadSpeed}) &\rightarrow \text{acceptable} \end{aligned}$$

Other preferences can be described similarly via the use of appropriate rules and will be introduced in detail later on.

### QoS Ontological Modeling

We assume that the user and the provider agree on a QoS upper ontology that represents the common knowledge in a specific application domain. This upper ontology can be refined by the user/provider to meet their requirements in terms of more detailed concepts and matching criteria. Because of the high complexity of the QoS information, we propose the use of a rule-based language that supports F-logic [10] to implement the QoS-related ontologies, instead of using DescriptionLogic as in other related work, e.g., [11].

Figure 1 shows the UML class diagram representation of the QoS upper ontology. QoSSpecification is the ontological concept corresponding to the QoS offering  $Q$  in the conceptual model. QoSParameter is the definition of the foundation quality concepts, such as RangeQuality, DownloadSpeed, ExecutionTime, etc., and their relationships. Users and providers can define their own domain-specific QoS concepts, e.g., AllowableDownloadSpeed, AverageExecutionTime, etc., by specializing the foundation ones. ContextualFactor defines the set of foundation contextual/environmental concepts such as InternetSpeed, Price, etc. that influence the other quality attributes. Other related concepts include the measurement methods (MeasurementModel) of a quality parameter, i.e., which quality attributes can be measured automatically or can only be estimated by human, and their corresponding metrics (MeasurementUnit).

ContextualDependency and QualityDependency represent the relations between a quality attribute value with its associated environmental conditions, and the dependencies among the QoS parameters themselves. For instance, in the file hosting scenario, ContextualDependency describes the relation between the offered UploadSpeed parameter with its associated contextual factors, which comprise the InternetSpeed of the user, the number of concurrent uploading files NoFilesUploading to guarantee the specified upload speed, and the Price a user has to pay for the service.

An important aspect of our formalism is the wide use of function symbols and rules to define various constraints, dependencies, matching and ranking preferences in this upper ontology (as well as in the derived ones). To check whether an offered quality value satisfies the user's requirements, we use the function QoSMatchingModel. The matching of contextual specification is similarly defined via the ContextualMatchingModel function. In fact, the rules implementing the ContextualMatchingModel and QoSMatchingModel functions are actually the ontological representation of the conceptual preferences  $P_k$  of a provider (or a user) described in Section 2. This ontological modeling enables the customization of both the QoS-based matching and ranking of services according to the preferences of the users and providers without changing the implementation code. Other modeled knowledge includes the personalized comparison between two quality values for the benefit of the ranking (QoSComparisonModel), and the conversion among the different measurement metrics, if possible (UnitChangingModel) and so forth. All of those above functions are implemented in the upper ontology and can be further customized in the derived ontologies.

Generally, our QoS conceptual model is simple yet comprehensive and compatible with most of the current standards and approaches. For example, our conceptual modeling of contextual description  $cmd$  can be interpreted as the combination of the Agreement Context, Expiration, and Qualifying Condition in the WS-Agreement specification [4]. Similar, our concept of quality constraint  $C'(qi)$  can be implemented to cover the notions of Agreement Creation Constraints, Agreement Offer, and Service level objective. The set of tuples  $\langle C'(qi), cmd \rangle$  of each semantic service description is equivalent to the notion of an Agreement Template in WS-Agreement or of a provider's policy in WS-Policy standard, etc. The use of F-logic and rule-based languages in our implementation of the conceptual model enables expressive descriptions of service's QoS advertisements for complex application scenarios. Furthermore, our work also includes various user's and provider's preferences into the conceptual model. The result is a powerful QoS modeling that should lead to a refined discovery process as we will show in the paper.

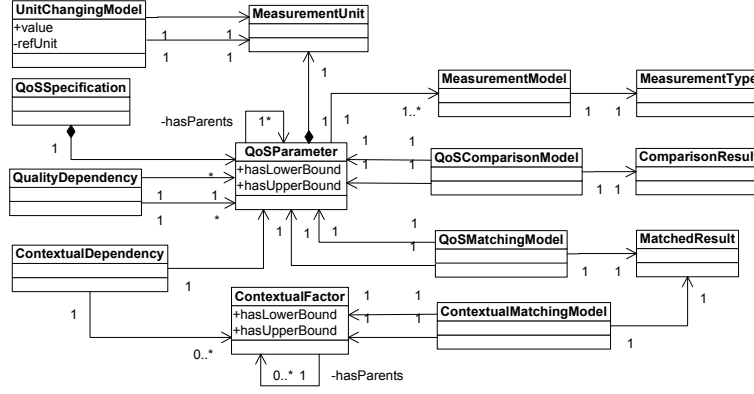


Figure 1: The QoS upper ontology.

### 3 Personalized QoS Matching Model

We only consider the matchmaking and ranking of services based on the QoS description part of the services, given that a set of services meeting the functional requirements has been obtained by other means already (e.g., via solution in Section 5). Given the presented conceptual model in Section 2, the matching between any complex QoS offering in a service  $s$  with a QoS requirement of query  $g$  can be decomposed into a set of basic matches  $\text{QoSMatching}(Q_g, Q_s)$  between a simplified QoS offering  $Q_s$  and a simplified requirement  $Q_g$ , where  $Q_s = \langle C'_s(qi_s), cnd_s \rangle$  only refers to one quality concept  $q_s$ , and  $Q_g = \langle C'_u(qi_u), cnd_u \rangle$  contains constraints over one concept  $q_u$ . Also, to ensure the decidability of the reasoning, we have to reduce some of the expressiveness of our conceptual modeling: (1) a user describes her execution environment  $cnd_u$  as a set  $C'_u$  of instances of related contextual factors; (2) a provider claims its offered QoS level  $C'_s$  as a quality instance  $qi_s$ . For convenient references, we summarize these notations in Table 1.

The personalized matching between a simplified QoS requirement  $Q_g$  and a simplified QoS offering  $Q_s$  is given in Algorithm 1. The matching function  $\mu_{ctx}$  is specified by the provider and needs to conform to the declaration of the `ContextualMatchingModel` (Figure 1). Symmetrically, a user describes in her goal the personalized QoS matching algorithm  $\mu_{qos}$  and the match result set  $M_{qos}$  that she accepts, in addition to her requirements  $C'_u$ .  $\mu_{qos}$  needs to follow the declaration of the function `QoSMatchingModel`. The sets  $M_{qos}$  and  $M_{ctx}$  contain instances of the concept `MatchedResult` in the QoS upper ontology. Default implementations of  $\mu_{ctx}$  and  $\mu_{qos}$  are available in the QoS upper ontology and thus both the provider and user can customize them flexibly in their own derived ontologies. For example,  $\mu_{qos}$  can be implemented as in the example

at the end of Section 2, where the user needs to define  $M_{qos} = \{\text{excellent, good, acceptable}\}$  as instances of the basic concept `MatchedResult` in the upper ontology.

More generally, a service  $s$  matches a user query  $g$  if all mandatory requirements of the user on different quality parameter  $q_u$  in the query  $g$  are satisfied by a certain simple QoS offering of a service, or  $\text{QoSMatching}(s, g) = m_{qos} \in M_{qos}, \forall q_u \in g$ . To conclude whether a service matches with a query or not, the discovery engine formulates the associated queries based on the declarations of the functions  $\mu_{ctx}$  and  $\mu_{qos}$ , which it knows about completely, and performs the reasoning on the constructed knowledge-base to find the matching result  $m_{qos}$ . Thus, the service matchmaking model is highly customizable both by the provider and the user via their own implementations of the functions  $\mu_{ctx}$  and  $\mu_{qos}$  in the domain QoS ontologies. Note that our algorithm does not impose any restriction in the constraint  $C'_u$  itself, i.e., depending on the capability of the reasoner,  $C'_u$  can be a complex logical expression on the properties of the required QoS instance  $qi_u$ . This also applies for the contextual constraints specified in  $cnd_u$ .

We suppose that the user and the provider may specialize the foundation QoS concept with further properties of their own. However, they would need to provide appropriate mediating rules to translate back and forth between the derived concept and the original one in the upper ontology. With such mediating rules, the reasoner would be able to detect whether the provider offers a quality parameter compatible with what the user expects, i.e., whether  $q_s$  and  $q_u$  are semantically-equivalent (line 9). For example, a service provider can combine the `DownloadSpeed` concept in the upper ontology with other domain-dependent concepts and business policies to represent its own quality attributes `MinDownloadSpeed`, `DownloadRate`, `AllowableDownloadSpeed`, etc. This additional knowledge is integrated into the knowledge-base and

Table 1: Frequently used notations

Provider-side		Client-side	
$Q_s$	a QoS offering by the provider	$Q_u$	a QoS requirement by the client (user)
$q_{i_s}^i$	provided quality instance (simplified form of $C_s^i(q_{i_s}^i)$ )	$C_u^i(q_{i_u}^i)$	required quality values, expressed as conditions $C_u^i$ on quality instances $q_{i_u}^i$
$cmd_s$	required environmental conditions (context)	$C_u$	set of instances describing user's context (simplified form of $cmd_u$ )
$P_s$	provider's preferences	$P_u$	user's preferences
$\mu_{ctx}$	matching rules to check if user's context is appropriate	$\mu_{qos}$	matching rules to check if provider's QoS is satisfactory
$M_{ctx}$	results of $\mu_{ctx}$ on which provider agrees to provide the service	$M_{qos}$	results of $\mu_{qos}$ client accepts as satisfactory to her requirements
$\mu_C$	pre-defined predicate to get results of $\mu_{ctx}$	$\mu_Q$	pre-defined predicate to get results of $\mu_{qos}$

can be reasoned about appropriately to detect that a provider also offers a DownloadSpeed at a certain level.

---

**Algorithm 1** QoSMatching( $Q_g, Q_s$ ) :  $m_{qos}$ 


---

- 1: Build the knowledge-base  $KB = Q_g \cup Q_s \cup$  related ontologies;
  - 2: Find matching function  $\mu_{ctx}$  in preferences  $P_s$ ;
  - 3: Bind all variables in  $cmd_s$  with values in  $C_u$ ;
  - 4: Obtain matching contexts  $X$  by querying the reasoner:  
 $KB \models \mu_C(X) := \mu_{ctx}(cmd_s, X)$ ;
  - 5: Return  $\perp$  if  $X$  is not in  $M_{ctx}$ ;
  - 6: Find the matching function  $\mu_{qos}$  in preferences  $P_u$ ;
  - 7: Obtain matching services  $Y$  by querying:  
 $KB \models \mu_Q(Y) := \mu_{qos}(C_u^i(q_{i_s}^i), Y)$ ;
  - 8: Return  $Y$  as the matching result;
- 

## 4 Personalized Service Ranking

Consider a user query  $g$  with QoS requirements  $\{\langle C_1^i(q_{i_1}), cmd_1, P_1 \rangle, \dots, \langle C_n^i(q_{i_n}), cmd_n, P_n \rangle\}$ , where all notations have the meanings as introduced in Section 2 and in Table 1. Suppose that the list of services that match the above query is  $L_g$ . For each  $S_i \in L_g$ , we define  $\widehat{q_{ik}}$ ,  $1 \leq k \leq n$  as the reputation-based QoS value of the QoS parameter  $q_{ik}$  provided by  $S_i$ , where  $q_{ik}$  is an instance of a QoS concept  $q_k$  in the ontology and  $\widehat{q_{ik}}$  is estimated as in our previous work [1, 2]. Since the evaluation of quality and the perception of the reputation information is subjective, user preferences and own judgements are relevant. Therefore, we include the following user preference information into the ranking procedure:

Firstly, a user may weight different quality parameters  $q_{ik}$  of service differently, e.g., she may state that the requirement on UploadSpeed is of lower importance than that of DownloadSpeed. We use  $w_k > 0$  ( $w_k$  can be a property of a QoSParameter concept) to denote the importance weight of the quality concept  $q_k$  to the user. Higher values of  $w_k$  mean the user considers  $q_k$  as more important and vice versa.

Secondly, the comparison between two quality values  $q_{ik}$  and  $q_{jk}$  of a QoS concept  $q_k$  is also important. We use the relation  $q_{ik} \succeq q_{jk}$  (resp.  $q_{ik} \prec q_{jk}$ ) to denote that the quality value  $q_{ik}$  is preferable (resp. less favored) than

the value  $q_{jk}$  by the user. This relation is specified via the QoSComparisonModel by the user in her preference ontology (derived from the upper QoS ontology). Note that this comparison should include the case where  $q_{ik}$  and/or  $q_{jk}$  does not exist in the descriptions of  $S_i$  and  $S_j$ .

Thirdly, each user may want to include the reputation-based estimated value  $\widehat{q_{ik}}$  into the rank computation differently, since each individual user have her own confidence on the credibility of the reputation mechanism, as well as on the sensitivity of reputation information to the actual value of different domain-dependent quality parameters. For instance, in the file hosting scenario, the DownloadSpeed offered by the service might be seen as more sensitive to its historical values than the SupportSize quality attribute since the latter is less likely to change. Thus we denote  $\alpha_k, 0 \leq \alpha_k \leq 1$  as the (common) subjective probability that the user trusts the advertised QoS of a provider, and  $\beta_k = 1 - \alpha_k$  as the probability that she believes in the reputation mechanism and thus in the estimation of  $\widehat{q_{ik}}$ . The quantity  $\beta_k$  (can be defined as a property of a QoSParameter concept) is used as a measure of confidence the user has on the reputation-based estimate value of that particular QoS parameter. Higher  $\beta_k$  values imply that: (1) the user has higher confidence in the reputation-based estimation  $\widehat{q_{ik}}$ , and (2) the user prefers the reputable services to the newly published ones. A user who wants to ignore the reputation value of a quality concept  $q_k$  simply sets  $\beta_k = 0.0$ .

The values of those above preferences can be provided by the user herself or defined by default in the upper or the derived QoS ontologies. This strategy enables the user to personalize the ranking as far as she wants, and the discovery solution is reusable for many different application domains without special knowledge about them.

The ranking of services based on their QoS properties is a multi-criteria decision problem, to which there are many possible solutions [12]. Here we employ a preference-based approach to develop our personalized ranking mechanism (Algorithm 2). We use the identity function  $1_P$  that evaluates to 1 if the predicate  $P$  is true and evaluates to 0 otherwise. Algorithm 2 ranks the services in  $L_g$  in the decreasing order of the probability that a user favors them (c.f. Propo-

sition 1 of [13]). An advantage of this method is its considerable genericness even for the case we do not know the ideal results for a certain query due to the complexity of the quality requirements  $C'_k(q_{ik})$ . The evaluation  $1_{\{q_{ik} \succeq q_{jk}\}}$  and  $1_{\{\widehat{q}_{ik} \succeq \widehat{q}_{jk}\}}$  can also be pre-computed to reduce the time cost of the discovery process.

---

**Algorithm 2** QoSRanking( $L_g$ ) : RankedList  $L_r$

---

```

1: for each  $S_i$  in  $L_g$  do
2:   for each  $S_j \neq S_i$  in  $L_g$  do
3:      $p_{ij} = \sum_k w_k \alpha_k 1_{\{q_{ik} \succeq q_{jk}\}} + w_k \beta_k 1_{\{\widehat{q}_{ik} \succeq \widehat{q}_{jk}\}}$ ;
4:   end for
5:    $P_i = \prod_{j \neq i} p_{ij}$ ;
6: end for
7: Return  $L_r$  as  $L_g$  sorted in the descending order of  $P_i$ 's;

```

---

## 5 Formal Model of the Discovery Process

One may envisage a single discovery component managing a large number of Web service descriptions and being targeted by numerous user queries with completely unpredictable arrival rates. In this context, the performance of a discovery process becomes of primordial importance as well as its ability to respond to variations on query arrival rates, while keeping the execution time of each query at an acceptable level. To provide such guarantees, we formally model the discovery process as a cost-based adaptive parallel query processing problem [14]. A query is modeled as an operator execution plan, in which nodes represent discovery operators and edges denote the dataflow between each pair of them. Potentially, a single discovery query may be modeled by a number of different operator execution plans, albeit equivalent in terms of the results they produce. Thus we derive an execution plan producing the smallest estimated cost for a given query.

We have identified a set of discovery operators that together form a discovery algebra (c.f. [15]). Each operator represents a particular function within the discovery process and may be implemented using different algorithms. For example, we have the following main operators:  $\mu_Q$  to match between a Web service description and a user query in term of QoS,  $\mu_F$  to assess the functional similarity between service description and a user query,  $\rho$  to rank services, and  $\theta$  to perform the evaluation of various QoS parameter values based on the ratings from the reputable users. In addition to ordering operators into an execution plan, our execution model extends traditional query execution by supporting reasonings and introducing some dynamic optimization techniques. The reasoning task is invoked as part of the execution of the  $\mu_Q$  and  $\mu_F$  operators and deserves special attention as it can become a bottleneck for the execution. Thus an efficient evaluation of a discovery query must target three main issues: (1) reduce the number of reasoning tasks;

(2) reduce the elapsed time for each individual Web service description semantic matchmaking evaluation; (3) adapt to variations in execution environment conditions. We cope with these three issues by introducing control operators into the execution plan that manage data transfer, data materialization, reasoning task parallelization and scheduling, etc. For brevity reasons, we refer the interested reader to our previous work [14] which describes the parallelization and adaptive execution strategies in detail.

Figure 2 illustrates a typical query execution plan (generated by the system) for processing general service discovery requests. Once a user query  $g$  and preferences  $a$  are entered, the scan operator  $\nu$  will read service descriptions  $s$  from the service repository and insert them into the query processing system. The execution process will be performed according to this plan via a parallel-pipelining processing mechanism. In this generated plan, there are a number of operators ( $\mu_F$  and  $\mu_Q$ ) being parallelized in order to reduce the total number of steps in processing a query. The operators  $\gamma$  and  $\circ$  in the query execution plan are automatically inserted by the system to handle the distribution of tasks and the collection of results.

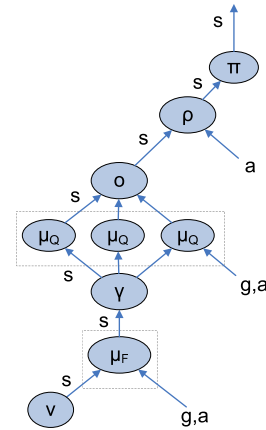


Figure 2: An example query execution plan.

## 6 Implementation and Experimentation

We implemented the prototype of our QoS-enabled discovery component using KAON2<sup>2</sup> as the reasoning engine and a WSMML-Flight reasoner wrapper<sup>3</sup> to translate from WSMML ontologies to KAON2's Datalog format. The adaptive discovery query processing system is developed from the existing implementation of the CoDISM-G framework [14]. Another third party light-weight functionality

<sup>2</sup><http://kaon2.semanticweb.org/>

<sup>3</sup><http://tools.deri.org/wsml2reasoner/>

discovery component [16] is used to perform the matching of services with a user goal by comparing their post-conditions. We also implement several ontologies based on real-world use cases using the WSML-Flight language, which covers a subset of F-logic [10]. These ontologies include the general purpose QoS upper ontology, the preferences and related ontologies for three use cases: the on-line file hosting, the hotel reservation, and the stock market broker application scenarios from one of our EU projects which has just finished [17]. For our online demonstration<sup>4</sup>, we develop a dedicated Web-based user interface to analyze the QoS-related ontologies, generate a GUI for user inputs and automatically formulates the ontology-based user preferences and goal descriptions for the discovery process.

Table 2 shows the results of an (illustrative) example service discovery query for a file hosting service offering DownloadSpeed higher than 25KB/s and UploadSpeed higher than 10KB/s. The user is willing to pay at most 10 Euros for her subscription and her Internet connection speed is ADSL 5Mbps. The service providers can specialize the DownloadSpeed concept in the upper ontology by defining various concepts MinDownloadSpeed, DownloadRate, AllowableDownloadSpeed, etc., with additional properties for their own uses. Thus, the use of semantics enables us to evaluate whether a syntactically-different but semantically-equivalent QoS parameter offered by a provider, e.g., DownloadRate, satisfies the user's requirements of DownloadSpeed or not.

The returned services satisfying both functionality and quality requirement of the user are  $S_1$  and  $S_2$ , in which  $S_1$  has a higher rank due to its higher reputation-based QoS ( $\widehat{q}_{11} \succeq \widehat{q}_{21}$  and  $\widehat{q}_{12} \succeq \widehat{q}_{22}$ ). Other services are rejected and removed from the discovery result since they either offer the quality level under those conditions that the user does not satisfy ( $S_3, S_4$ ) or since they do not satisfy the required functionality ( $S_5, S_6$ ). In general our discovery component can be used with more complicated scenarios where the QoS offers and requirements are of high complexity, e.g., a user may specify in her requirements that the statistics on the QoS parameter ExecutionTime of a candidate service (to be integrated in a Web service-based workflow management system) follows a certain distribution over different temporal periods for given input sizes. Similarly, providers can specify whatever prerequisites they want to impose on their potential clients, e.g., different prices according to different quality levels over time.

In our implemented prototype we limit ourselves to consider only exact matches between two QoS concepts during the matchmaking, assuming that we have a set of complete and correct translating rules between the derived concepts, e.g., MinDownloadSpeed, DownloadRate, AllowableDownloadSpeed etc. by a provider (or user) and

the original one DownloadSpeed in the upper ontology. As a result, the precision and recall of our ontology-based QoS discovery depends on the capability of the rule-based reasoner being used. The study of correctness of such reasoner is beyond the scope of this paper. The remaining relevant issue we need to analyze is the effectiveness of the ranking algorithm. Regarding this, our ranking approach exhibits the following properties.

- **One-time interaction:** The user provides her preferences only once before the discovery begins. These preferences are comprehensible and can be easily collected via the user interface. The whole matching and ranking processes are then done completely automatically, which is a benefit for users since the number of published services can be considerably high.
- **Informative results:** The services are ranked in decreasing order of the probability that a user favors them, taking into preferences of the users. This means the results are shown to the users in an appropriate way and can effectively support them to select their most favorite services.
- **Dominance detection:** Our ranking mechanism can detect the dominance among the services, i.e., if a service  $S_a$  is strictly better than  $S_b$  in term of a quality parameter  $q_k$ , and  $S_a$  is better than or equal to  $S_b$  in all other quality criteria, it is assured that  $S_a$  gets a higher rank than  $S_b$  in the final ranking result (c.f. Proposition 2 in [13]).

The reputation-based QoS estimation approach has been studied under various settings, which yields very accurate and reliable results even in highly vulnerable environments [1].

We also performed experiments running our service discovery engine using a parallel query processing system. The experiment objective was to evaluate the gains obtained by parallelizing operators of the discovery algebra. In particular, we execute in parallel a fragment of the query execution plan comprising the match ( $\mu_Q, \mu_F$ ) and the rank ( $\rho$ ) operators, in this order. We considered a repository containing 1000 web service descriptions synthetically generated. The execution environment comprises 20 homogeneous machines, one for the local operators and 19 parallel nodes. Figure 3 illustrates the obtained results. The values presented correspond to the average of five runs with the same configuration. One can observe that with 10 nodes, the overall response time is 3.6 times faster than the one obtained in the centralized execution. Note that this is the case considering the remote nodes initialization costs. In scenarios with larger number of remote machines (i.e., greater than 10 nodes for 1000 service descriptions), the gains obtained by parallelization start being blurred by initialization

<sup>4</sup><http://sirpeople.epfl.ch/thvu/download/qosdisc/>

Table 2: Example discovery result

	UploadSpeed( $q_1$ )	DownloadSpeed( $q_2$ )	Price	InternetSpeed	Result
<b>Requirements</b>	$\geq 10$ KB/s	$\geq 25$ KB/s	$\leq 10$ Euros	5Mbps	
<b>Preferences</b>	optional, $w_1 = 1, \beta_1 = 0.75$	optional, $w_2 = 2, \beta_2 = 0.75$			
FilesRUBasic( $S_1$ )	$\geq 100$ KB/s, $\widehat{q}_{11} = 102.9$ KB/s	$\geq 500$ KB/s, $\widehat{q}_{12} = 514.7$ KB/s	free	5Mbps	<b>rank=1</b>
UltraFiles4All( $S_2$ )	$\geq 10$ KB/s, $\widehat{q}_{21} = 10.5$ KB/s	$\geq 40$ KB/s, $\widehat{q}_{22} = 42.5$ KB/s	free	1Mbps	<b>rank=2</b>
FilesRUDeluxe( $S_3$ )	$\geq 100$ KB/s	$\geq 500$ KB/s	150CHF	5Mbps	<b>rejected</b> (advertised price too high)
UltraFilesPro( $S_4$ )	$\geq 100$ KB/s	$\geq 500$ KB/s	100CHF	10Mbps	<b>rejected</b> (advertised price too high, demanded Internet speed unavailable)
WSGetNewsXignite( $S_5$ )	-	-	100CHF	1Mbps	<b>rejected</b> (functionality unsatisfied)
ThemesHotel( $S_6$ )	-	-	-	-	<b>rejected</b> (functionality unsatisfied)

costs and remote node interferences. It is, however, interesting to observe that whenever the same query is executed twice, the whole initialization cost is hidden, enlarging the parallelization spectrum. We are currently enhancing our query optimizer to take into account the interference cost caused by remote nodes communicating with the local one. Our intention is to identify a break even point from which the initial input set should be split into multiple local nodes, keeping the interference under reasonable limits and allowing for greater parallelism.

Our modeling of the discovery process as a query execution plan also enables the plugging-in and comparison of the results of different variants of the reputation-based QoS estimation, matching, and ranking approaches. This will be addressed in our future work.

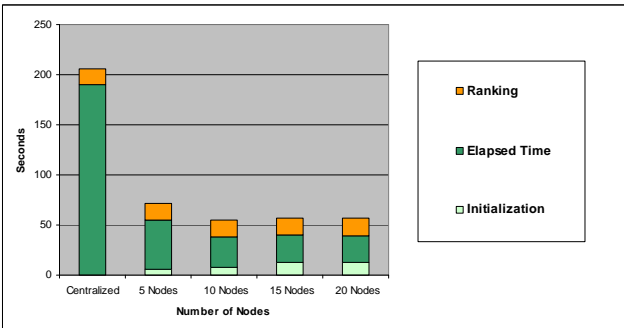


Figure 3: Experimental results

## 7 Related Work

Amongst the major efforts in using QoS criteria in service discovery are [18, 19] and [11]. The personalization of QoS-based service selection also interests a number of research work, e.g., Wagner et al. [20, 21] and [22]. Reputation information has also been partly used during the selection of services based on their QoS [23–26]. Due to limited

space, we refer the readers to the extended version of our paper [13] for a more extensive review. Herein, we summarize the difference between our solution with the most relevant work in Table 3 w.r.t the following dimensions:

- QEXP: the expressiveness of QoS model
- QEXT: the extensibility of the QoS model
- SWSE: whether the approach is semantic-enabled
- CXTE: whether the discovery is also based on checking of prerequisite conditions expressed by providers
- REPE: whether the approach employs reputation mechanisms to evaluate the trustworthiness of the advertised QoS
- PMCH: whether the matching algorithm is customizable (without changing the code)
- PRNK: whether the ranking algorithm can be personalized w.r.t. user preferences
- FLEX: the possibility of integrating different algorithms during the discovery, e.g., using different reputation mechanisms to estimate services' quality
- OPTe: the easy parallelization and optimization of the whole discovery process

A  $\checkmark$  in Table 3 denotes that the corresponding feature is supported and a  $*$  implies that the issue is (partially) addressed by some work in the mentioned group.

## 8 Conclusions

We have presented an approach for discovery of services that enables both functional- and quality-based discovery. The proposed discovery framework is highly extensible and customizable, which adequately addresses many relevant issues in the literature: semantic modeling of QoS, personalized matchmaking and ranking of services, and the use of services' QoS reputation in the discovery process. Additionally, the important steps of the discovery process are implemented as operators in a discovery algebra, making our approach customizable and scalable through implicit parallelization capabilities.

Table 3: Comparison of our framework with others

	Ours	[18]	[19]	[11]	[20,21]	[22]	[24]	[23,25,26]
QEXP	✓	✓	✓	✓		✓		*
QEXT	✓	✓	✓	✓	✓	✓	✓	*
SWSE	✓	✓	✓	✓	✓	✓		*
CXTE	✓	✓	✓	✓				
REPE	✓						✓	✓
PMCH	✓	✓			✓	✓		
PRNK	✓	✓			✓	✓	✓	*
FLEX	✓							
OPTE	✓							

## References

- [1] L.-H. Vu, M. Hauswirth, and K. Aberer, "QoS-based service selection and ranking with trust and reputation management," in *Proceedings of the Cooperative Information System Conference (CoopIS'05)*, pp. 446–483, 2005.
- [2] L.-H. Vu, M. Hauswirth, F. Porto, and K. Aberer, "A search engine for QoS-enabled discovery of Semantic Web services," *International Journal of Business Process Integration and Management*, 1(3): 244–255, 2006.
- [3] S. B. et al, *Web Services Policy Framework*. <http://www.w3.org/Submission/WS-Policy/>, 2006.
- [4] A. Andrieux et al, *Web Services Agreement Specification (WS-Agreement) Version 2005/09*. <http://www.w3.org/Submission/WS-Policy/>, 2005.
- [5] H. Ludwig, A. Keller, A. Dan, R.-P. King, and R. Franck, *Web Service Level Agreement (WSLA) Language Specification*. <http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf>, 2003.
- [6] S. Frolund and J. Koisten, *QML: A Language for Quality of Service Specification*. <http://www.hpl.hp.com/techreports/98/HPL-98-10.html>, 1998.
- [7] V. Tasic, *Service Offerings for XML Web Services and Their Management Applications*. PhD thesis, Department of Systems and Computer Engineering, Carleton University, Canada, 2004.
- [8] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. <http://www.w3.org/Submission/SWRL/>.
- [9] *D2v1.3. Web Service Modeling Ontology (WSMO)*. <http://www.wsmo.org/TR/d2/v1.3/>.
- [10] M. Kifer, G. Lausen, and J. Wu, "Logical foundations of object-oriented and frame-based languages," *J. ACM*, 42(4): 741–843, 1995.
- [11] C. Zhou, L.-T. Chia, and B.-S. Lee, "Web services discovery with daml-qos ontology," *Int. Journal of Web Services Research (JWSR)*, 2(2): 44–67, 2005.
- [12] F. Naumann, "Data fusion and data quality," in *Seminar on New Techniques and Technologies for Statistics*, (Sorrento, Italy), 1998.
- [13] L.-H. Vu, F. Porto, M. Hauswirth, and K. Aberer, "An Extensible and Personalized Approach to QoS-enabled Semantic Web Service Discovery," Tech. Rep. LSIR-REPORT-2006-012, EPFL, 2006, available at <http://infoscience.epfl.ch/search.py?recid=89160>.
- [14] F. Porto, V. F. V. da Silva, M. L. Dutra, and B. Schulze, "An adaptive distributed query processing Grid service," in *Proceedings of the Workshop on Data Management in Grids, VLDB*, 2005.
- [15] M. Hauswirth, F. Porto, and L.-H. Vu, *P2P and QoS-enabled service discovery specification. DIP Project Deliverable D4.17*, available from <http://dip.semanticweb.org/documents/D4.17-Revised.pdf>, 2005.
- [16] A. Friesen and S. Grimm, *SWS Discovery Module Specification. DIP Project Deliverable D4.8*. <http://dip.semanticweb.org/documents/D4.8Final.pdf>.
- [17] *DIP Integrated project- Data, Information, and Process Integration with Semantic Web Services*. <http://dip.semanticweb.org/>.
- [18] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour, "Semantic WS-agreement partner selection," in *Proceedings of WWW'06*, pp. 697–706, 2006.
- [19] N. Sriharee, T. Senivongse, K. Verma, and A. P. Sheth, "On using ws-policy, ontology, and rule reasoning to discover web services.," in *Proceedings of INTELCCOMM*, pp. 246–255, 2004.
- [20] W.-T. Balke and M. Wagner, "Through different eyes: assessing multiple conceptual views for querying web services," in *Proceedings of WWW'04*, pp. 196–205, ACM Press, 2004.
- [21] W.-T. Balke and M. Wagner, "Towards personalized selection of web services.," in *Proceedings of WWW'03*, 2003.
- [22] T. D. Noia, E. D. Sciascio, F. M. Donini, and M. Mongiello, "A system for principled matchmaking in an electronic marketplace," in *Proceedings of WWW'03*, pp. 321–330, 2003.
- [23] A. S. Bilgin and M. P. Singh, "A DAML-based repository for QoS-aware semantic web service selection," in *Proceedings of ICWS'04*, pp. 368–375, 2004.
- [24] Y. Liu, A. Ngu, and L. Zheng, "QoS computation and policing in dynamic web service selection," in *Proceedings of WWW'04*, pp. 66–73, 2004.
- [25] S. Ran, "A model for Web services discovery with QoS," *SIGecom Exch.*, 4(1): 1–10, 2003.
- [26] M. Ouzzani and A. Bouguettaya, "Efficient access to Web services," *IEEE Internet Computing*, pp. 34–44, March/April 2004.