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

Le-Hung Vu, Fabio Porto, Karl Aberer
Swiss Federal Institute of Technology Lausanne
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 a framework for the autonomous discovery and selection of Semantic Web services based on their QoS properties. The novelty of our approach is the wide use of semantic technologies for a customizable discovery, which enables both the service users and providers to flexibly specify their matching models for QoS and the corresponding environmental conditions. In the presented approach, the discovery and ranking of services can be personalized via the use of domain ontologies detailing the user's preferences and the provider's specification. The discovery component 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, which facilitates the introduction of different discovery algorithms and the automatic generation of appropriate parallelized matchmaking evaluations, enabling the scalability of our solution up to unpredictable arrival rate of user queries against high numbers of published service descriptions in the system.

1 Introduction

Currently majority of services and information on the Internet are only available in the form of human-readable Web pages, thus requiring human intervention in order to discover and execute appropriate services necessary to achieve a certain user's goal. In the vision of the Semantic Web and given the support of the emerging Web service technologies, the wrapping of such services in the form of Semantic Web services would make them self-described and widely accessible both for human users and autonomous agents [10].

In this paper, we consider the scenario where many providers offering a variety of services, each of which can represent a typical business activity such as an online hotel reservation, a book shipping, a pizza ordering service or a Web-based software component like stock market information or matrix computational services, etc. Thus, a Semantic Web service is an electronic description of certain concrete services in reality, i.e., each real-life service is assumed to be semantically annotated and registered (published) to a service registry in the form of a Semantic Web service. Such a registry provides users the facilities to search for services with certain properties.

In the context of autonomous service usage, an important aspect is to allow agents to discover those services fulfilling the requirements of a user in terms of both the functionality and the non-functional properties. In fact, some non-functional requirements maybe as relevant as the required functionality itself, as it is the case of Quality of Service (QoS). QoS is usually the decisive criterion a user considers 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's Web sites offering similar capabilities in terms of hotel online booking service, a user would lean towards the one with the shorter reservation response, offering better price conditions, more comfortable rooms and higher customer-care facilities. Several other service instances may be considered to have QoS features as their main differentiating critera, such as file sharing/hosting, online TV/radio station and online music store, teleconference, photo sharing/exchanging services, etc.

Different from the discovery of services matching functional requirements, we claim that the discovery of services based on their QoS is more complicated and the *personalization* of the QoS-based discovery process to adapt with different needs of the users is a strong requirement. Since we consider a Web service description as an electronic advertisement of real-life services, it can include

*This work was (partly) funded by 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 Líon project funded by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

many domain-dependent QoS properties. Such quality information is dynamic and depends on many factors: other quality parameters and the related user-side contextual or environmental conditions. Therefore, the advertisement of quality in a service description should only be considered as a claim, which the provider engages to offer under certain conditions and which should be verified and validated over time. This important evaluation of the reputation of the services is a subjective process, since different users have different interpretations of the reputation of a certain service. Secondly, the suitability of the service to a requirement of the user in terms of a QoS criterion is also subject to her[1] preferences. For example, the conclusion whether a certain deviation of quality is still acceptable should only be determined by the user. Moreover, the search results should reflect the ranking of the matched services most appropriately according to the user's objectives and criteria, 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. QoS. The goal of the QoS-enabled service discovery process in our work is to automatically find those service descriptions which match the requirements of the users both in terms of functionalities and QoS, letting the actual negotiation, selection, and execution of the service be done either electronically or in reality depending on the nature of services. Extending our previous work in [44, 45], we introduce here a complete discovery solution which combines our previously developed techniques and widely exploits the semantic technologies to enable the personalization of the whole QoS-enabled discovery process. We validate this with an implemented prototype and provide corresponding experimental results. The contribution of our approach is its *high extensibility and customizability* of the discovery, ranking, and selection via the use of domain ontologies detailing the user's preferences and provider's specification. Specifically, our approach has the following advantages:

- **Expressive and extensible conceptual modeling of service QoS** Given the above complexity and dynamics of QoS information, we propose an adequate semantic conceptual modeling approach for the flexible specification of user's requirements and the QoS offerings of service, which is simple yet expressive and be compatible with most of the current standards and approaches [15, 20, 22, 29, 43].

- **Customizable matchmaking model** Via appropriate exploitation of semantic technologies, especially rule-based languages and reasoners, we can express the QoS requirements, advertisements, user's matching conditions and preferences flexibly. The QoS-enabled discovery process can be done autonomously and efficiently by reasoning on the constructed knowledge-bases based on the various personalized matching criteria and preferences of the users.

- **Personalized ranking model** To provide useful and informative ranking results supporting the user in the selection of the most appropriate services to execute, we consider various important information of the different quality dimensions of the services, their reputation, as well as variety of preferences from the users.

- **Flexible and scalable implementation** We model the whole discovery engine 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 modeling enables us to apply cost-based optimization strategies to parallelize the evaluation of the expensive operators, considering that there can be unpredictable service search queries from many users and that the number of published Web service descriptions may substantially increase in the future. Moreover, it facilitates the *plugging-in, testing, and comparison* of different algorithms on-the-fly.

- **Prototype already available** Our implemented prototype validates our approach and confirms the usefulness of semantic technologies in dealing with the above issues nicely. Our prototype as well as the online demonstration, related ontologies, and documentation are freely available on the Web[2] and can be an interesting case study of how Semantic Web technologies can be exploited in real-world applications. We adopt the WSMO ontology framework in our implementation as a proof-of-concept of our work. However, it is generally applicable to other models, such as OWL-S+SWRL [5].

Besides the above contributions, we also include an effective reputation-based QoS estimation to accurately evaluate the QoS parameter values for available services given their historical data collected from various information sources: the users, the providers' advertisement, and the ratings from a few trusted agents.

The rest of the paper is organized as follows: in Section 2 we present the proposed semantic QoS conceptual model and associated examples. Our personalized discovery algorithms are described in details in Section 3. Section 4 provides the description of our discovery prototype

[1] Generally a user in this paper can represent either a person, an autonomous agent, or another system. For brevity, we simply use the feminine personality to refer to a specific user henceforth.

[2] http://lsirpeople.epfl.ch/lhvu/download/qosdisc/

and our analytical and experimental results. We review the related work in Section 5 and finally conclude the paper in Section 6.

2 Semantic Modeling of QoS

2.1 Conceptual Modeling

Our approach for describing Web services builds on the fact that a consumer analyzes and selects a service according to certain criteria, the most common one being the functionality and the quality offered by the service. As a typical example, consider an online file hosting service whose function is to offer users file storage management facilities. The functionality specified by this Web service describes that it must have the capability of uploading and downloading files, i.e., this is part of the functional service description. In contrast, the criterion we are mainly interested in is its QoS, i.e., quality-related parameters included in the semantic description of the service advertisement and the user requests, for example, the download and upload speed that this file hosting service offers, the maximally allowed size of the file, the number of concurrent downloads/uploads, the maximal duration that the server agrees to store the file, etc.

In this context, selecting a service comprises finding those services offering the desired functionality and the ones which fulfill the quality requirements of the user. Therefore, in our model, Web services are described by conjunctive sets of properties $\mathsf{F} \wedge \mathsf{Q}$, where F is the functional description and Q is the QoS offering description part. In this work, we focus on the modeling of a service's QoS capability, reusing the functionality description as specified in the WSMO model [16]. 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. We do not focus on the negotiation between the service provider and a user, which we consider as a later step after the user already obtains the results from the discovery. Based on this, our model for describing a QoS offering of a service includes the following aspects:

- The semantic description of each QoS parameter and its corresponding quality level offered by a certain service provider, e.g., parameter names and textual descriptions, possible values of the parameters, respective measurement units, associated evaluation methods, etc. We model this part as $C'(qi)$, in which C' is a concept expression that constrains the instance qi of a QoS concept in the QoS domain ontology.
- The necessary and sufficient environmental (or contextual) conditions that a user shall agree to so that the provider can guarantee the offered quality levels. Examples of environment conditions include: a minimum Internet connection speed, the user's location, the number and input size of requests (to satisfy requirement on execution-time and response-time), third parties engagements, etc. Environmental conditions are expressed as an axiom cnd over instances of a set of environment concepts.
- The preferences of the provider in selecting a user. More specifically, this is a set of rules P determining the acceptable matching levels between a specific contextual setting of a requester and each constraint in cnd imposed by the service.

Generally, we describe a QoS offering Q in the service description as a set $\{\langle C'_1(qi_1), cnd_1, P_1\rangle, \ldots, \langle C'_n(qi_n), cnd_n, P_n\rangle\}$, 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 over instances of those concepts 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 $C'_k(qi_k) = \{\mathsf{uploadSpeed} \geq \mathsf{100KBps}\}$ as the average upload speed that it offers, and $cnd_k = \{\mathsf{internetSpeed} \geq \mathsf{1Mbps}, \mathsf{noFilesUploading} = 1, \mathsf{price} = \mathsf{10\$}\}$ 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.

$$\mathsf{userPrice} \in \mathsf{Price} \wedge \mathsf{userPrice} \geq \mathsf{price} \rightarrow \mathsf{priortyClient}$$
$$\mathsf{userPrice} \in \mathsf{Price} \wedge \mathsf{userPrice} < \mathsf{price} \rightarrow \mathsf{acceptedClient}$$

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

Symmetric to a Web service description, a service discovery request (also called a user query, a service query, or a user's goal in the descriptions to follow) consists of the description of the functionality and the QoS a Web service should offer to fulfill a user's needs. A user query is also specified as $\mathsf{F} \wedge \mathsf{Q}$, where F is the functional requirement description and Q is the QoS requirement specification.

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., $C'_k = \{\mathsf{reqUploadSpeed} \geq \mathsf{20KBps}\}$. This QoS requirement is complemented by the contextual conditions

cnd_k the client is able to agree with, e.g., $cnd_k = \{\mathsf{userInternetSpeed} = \mathsf{adsl1Mbps},\ \mathsf{noFilesUploading} = 1,\ \mathsf{userPrice} = 15\$\}$. The set of preferences rules P_k in the goal model comprises various settings for the discovery process: which quality parameters are preferred by the users, how much the user trusts the reputation information on a service, the matching levels for the QoS required by the user, 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 this service (with a quality parameter qs) satisfies her requirement. The following rules describe these preferences:

$$\mathsf{qs} \in \mathsf{ReqUploadSpeed} \wedge \mathsf{qs} \geq 100\mathsf{KBps} \rightarrow \mathsf{excellent}$$
$$\mathsf{qs} \in \mathsf{ReqUploadSpeed} \wedge \mathsf{reqUploadSpeed} \leq \mathsf{qs} < 100\mathsf{KBps} \rightarrow \mathsf{good}$$
$$\mathsf{qs} \in \mathsf{ReqUploadSpeed} \wedge \mathsf{qs} < \mathsf{reqUploadSpeed} \rightarrow \mathsf{acceptable}$$
$$\forall \mathsf{qs} \neg(\mathsf{qs} \in \mathsf{ReqUploadSpeed}) \rightarrow \mathsf{acceptable}$$

The other preferences, which will be introduced in detail later on, can be described similarly via the use of appropriate rules.

The above conceptual model is simple yet powerful, once implemented with an appropriate ontological framework, such as WSMO [16], and combined with dedicated semantic technologies supporting the rule-based reasonings, e.g., KAON2 reasoner[3]. This is sufficient for the specification of sophisticated QoS requirements and offerings in various realistic application scenarios, as we will show in the following sections.

2.2 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 [27] to implement the QoS-related ontologies, instead of using Description Logic as in other related work, e.g., [47].

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 for 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 2.1. These 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 by default in the upper ontology and can be overwritten in the derived ontologies.

We have implemented these QoS-related ontologies for various realistic applications, e.g., the file hosting, the hotel reservation, and the stock-market information service use cases using the WSML – Flight [1] language, a subset of F-logic. Due to space limitation, we point the interested readers to the download page of our component[4] for further details.

[3] http://kaon2.semanticweb.org

[4] http://lsirpeople.epfl.ch/lhvu/download/qosdisc/

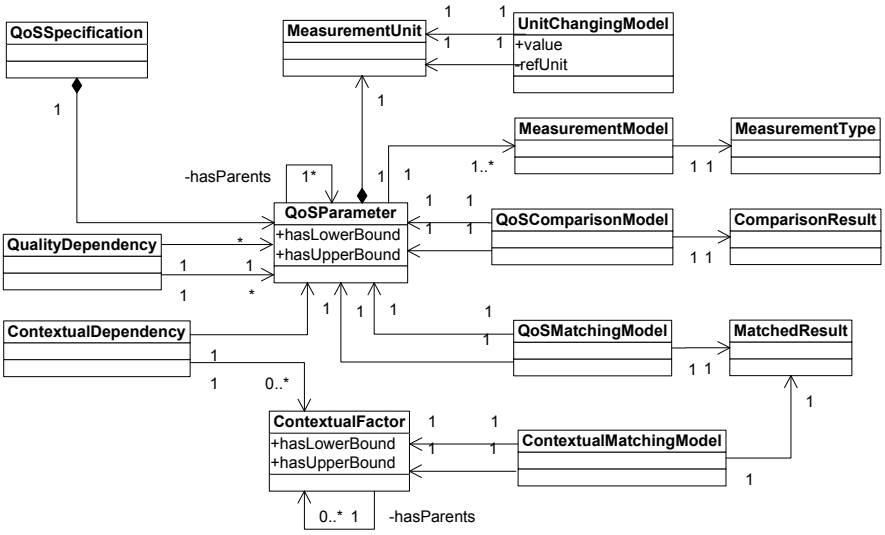

Figure 1. The QoS upper ontology.

3 Solution Approach

3.1 Personalized QoS Matching Model

The QoS semantic modeling in Section 2.1 leads to a symmetric representation of the Web service description and the user query, expressed as $\mathsf{F} \wedge \mathsf{Q}$. Nevertheless, F and Q have different meanings. As a result, the semantic matchmaking follows different models when considering functional and QoS properties. In this paper, we only consider the matchmaking and ranking of services based on the QoS description part of the services, given that a set of services with similar functionality has been obtained via another algorithm (see section 3.4 for our solution of integrating the results of these two algorithms).

Given the conceptual model in Section 2.1, the matching between any QoS offering in a service s with a query g can be decomposed into a set of matches among many QoS offerings Q_s with different QoS requirements 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. 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 those contextual factors influencing the values of the quality parameters in the query; (2) a provider claims its offered QoS level C'_s as a quality instance qi_s.

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.

The personalized matching between a QoS requirement Q_g and a QoS offering Q_s is given in Algorithm 1. μ_C and μ_Q are our pre-defined predicates to obtain the matching results X and Y by querying the DataLog knowledge-base KB. μ_{ctx} is a set of rules to determine whether the environment of the user satisfies the prerequisites of the provider, and M_{ctx} are the set of results that a service provider accepts to provide the QoS level C'_s. μ_{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 μ_{qos} and the match result set M_{qos} that she accepts, in addition to her requirements C'_u. μ_{qos} needs to follow the declaration of the function QoSMatchingModel. Elements of the acceptable result sets M_{qos} and M_{ctx} are instances of the concept MatchedResult in the QoS upper ontology. The default implementations of μ_{ctx} and μ_{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, μ_{qos} can be implemented as in the example at the end of Section 2.1, where the user needs to define $M_{qos} = \{\mathsf{excellent}, \mathsf{good}, \mathsf{acceptable}\}$ as instances of the basic concept MatchedResult in the upper ontology.

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 of-

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

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

INPUT:
$Q_s = \langle qi_s, cnd_s, P_s \rangle$;
$Q_g = \langle C'_u(qi_u), C_u, P_u \rangle$;
1: Build the knowledge-base $KB = Q_g \cup Q_s \cup$ related ontologies;
2: Find the matching function μ_{ctx} in preferences P_s;
3: Bind all variables in cnd_s with values in C_u;
4: Submit query $KB \models \mu_C(X) := \mu_{ctx}(cnd_s, X)$ to the reasoner;
5: **if** X is not in M_{ctx} **then**
6: Return $\perp$; /*$\perp$ is not in M_{qos}*/
7: **end if**
8: Find the matching function μ_{qos} in preferences P_u;
9: Submit query $KB \models \mu_Q(Y) := \mu_{qos}(C'_u(qi_s), Y)$;
10: **if** $Y \in M_{qos}$ **then**
11: Return Y;
12: **else**
13: Return $\perp$;
14: **end if**

More generally, a service s matches a user query g if all 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 $\mathsf{QoSMatching(s, g)} = m_{qos} \in M_{qos}$, $\forall q_u \in g$. 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 μ_{ctx} and μ_{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 μ_{ctx} and μ_{qos} in the domain QoS ontologies.

3.2 Personalized QoS-based Service Ranking

Consider a user query g with QoS requirements $\{\langle C'_1(qi_1), cnd_1, P_1 \rangle, \ldots \langle C'_n(qi_n), cnd_n, P_n \rangle\}$, where $C'_k(qi_k)$, $1 \leq k \leq n$, represents the required QoS level of a QoS concept q_k in a QoS ontology, and cnd_k is the user's associated context to achieve C'_k, respectively. Suppose that the list of services that match the above query is $L_g = \{S_1, S_2, \ldots, S_m\}$. For each $S_i \in L_g$, we define $\widehat{q_{ik}}$ as the reputation-based QoS value of the QoS parameter q_{ik} provided by S_i. We estimate $\widehat{q_{ik}}$ as described in our previous work [44, 45].

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, since each service may exhibit many quality parameters q_{ik}, a user may weight these parameters 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 defined as 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, a user may need to define the ordering between two quality values q_{ik} and q_{jk} of a QoS concept q_k. 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, as 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 considered 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 β_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 β_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, each of these is suitable with a certain domain and with certain properties of data sources [32]. Here we employ a preference-based approach to develop our personalized ranking mechanism (Algorithm 2), i.e. we give higher ranks to services that are more likely to be preferred to the user. An advantage of this method is its considerable genericness even for the case we do not know the ideal results for a certain query to the complexity of the quality requirements $C'_k(qi_k)$. Proposition 1 explains the rationales behind this ranking algorithm. For brevity, we use the identity function 1_P that evaluates to 1 if the predicate P is true and evaluates to 0 otherwise. 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($\mathsf{L_g}$) : RankedList$\mathsf{L_r}$

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;

Proposition 1. *Algorithm 2 ranks the services in L_g in the decreasing order of the subjective probability that a user favors them.*

Proof. Let $S_i \succeq S_j$ (resp. $S_i \prec S_j$) be the event that a user prefers (or resp. does not favor) S_i to S_j with respect to her preferences. Additional, define $S_i \succeq^k S_j$ as the event that the user favors S_j than S_j in terms of the quality dimension q_k. Given the weight w_k denoting the importance of each quality concept q_k to the user, the probability that q_k is a decisive factor in the service selection of the user is $w_k / \sum_{w_k}$. Therefore:

$$\begin{aligned} Pr(S_i \succeq S_j) &\propto \sum_k w_k Pr(S_i \succeq^k S_j) \\ &\propto \sum_k w_k \alpha_k 1_{\{q_{ik} \succeq q_{jk}\}} \\ &\quad + w_k \beta_k 1_{\{\widehat{q_{ik}} \succeq \widehat{q_{jk}}\}}) \\ &= p_{ij} \end{aligned}$$

Similarly, we have:

$$\begin{aligned} Pr(S_i \prec S_j) &\propto \sum_k w_k \alpha_k 1_{\{q_{ik} \prec q_{jk}\}} \\ &\quad + w_k \beta_k 1_{\{\widehat{q_{ik}} \prec \widehat{q_{jk}}\}}) \\ &= q_{ij} \end{aligned}$$

The probability that the user favors the service S_i than all other services is:

$$\begin{aligned} Pr(S_i \succeq S_j, \forall j \neq i) &= Pr(\cap_{j \neq i} S_i \succeq S_j) \\ &= \prod_{j \neq i} Pr(S_i \succeq S_j) \\ &= \prod_{j \neq i} p_{ij}/(p_{ij} + q_{ij}) \\ &= \prod_{j \neq i} (p_{ij} / \sum_k w_k) = P_i / (\sum_k w_k)^{m-1} \end{aligned}$$

The last two equalities are due to the fact that: given any two matching levels x, y, we have $1_{\{x \succeq y\}} + 1_{\{x \prec y\}} = 1$, leading to $p_{ij} + q_{ij} = \sum_k w_k$. □

3.3 Reputation-based QoS Assessment

The discovery of services in terms of QoS requires an accurate evaluation of how well a service can fulfill a user's quality requirements. For this estimation, we use a reputation-based model which exploits data from many information sources: (1) We use the QoS values promised by providers in their service advertisements; (2) we provide an interface for the service users to submit their feedback on the perceived QoS of consumed services; (3) we also use similar reports produced by a few trustworthy QoS monitoring agents, e.g., rating agencies.

For each QoS parameter q_k offered by a service S, we evaluate the real capability of this service in providing this QoS parameter to the users as follows: With every context cnd_{kj}, i.e., a set of real-world conditions as in Section 2.1, in which the service S advertises q_k, the related QoS instances q_{kjt} are collected at different time t. In other words, we gather all user ratings which are on S and refer to q_k in the corresponding context cnd_{kj}. Such a rating by a user u at time t has the form $\langle u, S, t, q_{kjt} \rangle$ where q_{kjt} is reported under the context $ucnd_{kjt}$ which matches with the context cnd_{kj} required by the provider. After doing the analysis and filtering out unreliable reports, the reputation-based estimation of the actual quality value of q_k in context cnd_{kj} is an instance $\widehat{q_{kj}}$ computed as follows: Since each QoS instance q_{kjt} consists of a list of property-value pairs $\langle p_l, v_{lt} \rangle$, each reputation-sensitive property p_l of $\widehat{q}_{kj}$ would have the value $\widehat{v}_l$. The estimation of a $\widehat{v}_l$ based on its historical statistics $\langle t, v_{lt} \rangle$ is then done using the time-based regression methods which we proposed in [44].

For the QoS estimation, we only choose the most reliable reports from the reputable raters. This is done via a collaboratively and statistical approach based on two realistic assumptions. First, we assume *probabilistic behavior of services and users*, meaning that the differences between the real quality conformance which users obtained and the QoS values they report follow certain probability distributions. These differences vary depending on whether users are honest or cheating as well as on the level of changes in their behaviors. Secondly, we presume that *there exist a few trusted rating agencies*. These agents always produce credible QoS

reports and are used as trustworthy information sources to evaluate the behaviors of the other users. In reality, a company managing the discovery component can deploy special applications to obtain the statistics on QoS of some specific Web services. Alternatively, it can also hire another third party companies to do these monitoring tasks. In order to detect possible frauds in user feedbacks, we use reports of trusted agents as reference values to evaluate behaviors of other users by applying a trust-distrust propagation method and a clustering algorithm. Reports that are considered as incredible will not be used in the QoS evaluation process. Note that this reputation-based QoS estimation can be done incrementally and in off-line mode, thereby it does not affect to the system's overall performance. For further details we refer to readers to our previous work [44].

3.4 Formal Modeling of the Service 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 incoming query arrival rates, while keeping the discovery time of each query at an acceptable level.

In order to provide such guarantees, we model the discovery process as a cost-based adaptive parallel query processing problem [38]. Within the discovery process we distinguish independent operators with clear semantics and to which estimated evaluation costs may be associated. A discovery 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 execution plans, albeit equivalent in terms of the results they produce. Thus we derive a plan producing the smallest estimated cost for a given discovery query.

We have identified a set of discovery operators that together form a discovery algebra (see Section 2.4.3, page 24 of [23] for a more detailed description). Each operator represents a particular function within the discovery process and may be implemented using different algorithms.

- *restriction* δ - reduces the set of Web service description candidates for matching with the user query via the use of Bloom key as described in our previous work [45].
- *QoS match* μ_Q - applies a semantic matchmaking algorithm to assess the similarity between a Web service description and a user query in term of QoS. Matchmaking is implemented as described in Section 3.1.
- *functionality match* μ_F - applies a semantic matchmaking algorithm to assess the similarity between a Web service description and a user query in term of its functionality. Herein, we suppose to use any existing implementation available to get a list of functionally equivalent list of services.
- *rank* ρ - orders matched Web service descriptions based on the results of the match operation and according to the user's preferences, as discussed in Section 3.2.
- *reputation-based QoS estimation* θ - performs the evaluation of various QoS parameter values based on the ratings from the reputable users, as explained in Section 3.3.

Other pre-defined operators include: (1) the *split*/*merge* operators $\gamma/\circ$ to supports parallelization by distributing and merging individual tasks/results to another processing node; (2) the *scan* operator ν to read Web service descriptions from a repository and format them as input tuples; (3) the *project* operator π to select the results and send back to the user.

In addition to ordering operators into an operator execution plan, our execution model extends traditional query execution by supporting reasoning and introducing some dynamic optimization techniques. The reasoning task is invoked as part of the *QoS match* and *functionality match* 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 operator 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 [38] which describes the parallelization and adaptive execution strategies in detail.

Figure 2 illustrates a typical operator execution plan (generated by the system) for processing general service discovery requests. Once a user query g and preferences a are entered, the scan operators ν will read service descriptions s from the service repository and feed them one by one 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 be a number of operators being parallelized (these μ_F and μ_Q operators) in order to reduce the total number of processing a service query. Note the operators γ and $\circ$ in the query execution plan are automatically inserted by the system to handle the distribution of tasks and collection of results.

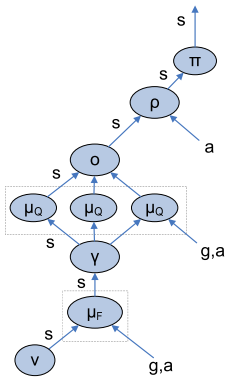

Figure 2. An example query execution plan.

4 Implementation and Experimentation

We implement the prototype of our QoS-enabled discovery component using KAON2[5] as the reasoning engine and a WSML-Flight reasoner wrapper[6] to translate from WSML ontologies to KAON2's Datalog format. The adaptive discovery query processing system is developed from the existing implementation of the CoDISM-G framework [38] with the addition of new discovery algebraic operators: QoS match, ranking, reputation-based QoS estimation, and Bloom filtering. We also use another third party lightweight functionality discovery component, which performs the matching of services with a user goal by comparing their post-conditions [21]. We also implement several useful ontologies using the WSML-Flight language, which covers a subset of F-logic [27]. These ontologies include the general purpose QoS upper ontology, the preferences and related ontologies for three use cases: the online file hosting, the hotel reservation, and the stock market broker application scenarios from one of our projects [2]. For our online demonstration[7], we use the Google Web Toolkit[8] to develop a dedicated Web-based user interface which analyzes the QoS-related ontologies, generates appropriate GUI to ask for user inputs and automatically formulate the ontology-based user preferences and goal descriptions for the discovery process.

Table 1 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. However, they would need to provide appropriate mediating rules to translate between the derived concepts with the original DownloadSpeed concept. Thus, the use of semantics herein 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 return 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 better reputation ($\widehat{q_{11}} \succ \widehat{q_{21}}$ and $\widehat{q_{12}} \succ \widehat{q_{22}}$). Other services are rejected out of the discovery result since either they offer the quality level under those conditions that the user does not satisfy (S_3, S_4) or they do not offer the required functionality (S_5, S_6). Note that the above example is only for demonstration purpose. 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.

We observe the following nice properties of our selection and ranking mechanism.

- **One-time interaction** The user provides her preferences only once before the discovery begins. These preferences are comprehensible and can be easily collected via appropriate user interface. The whole matching and ranking process are then done automatically.
- **Informative results** The services are ranked in the decreasing order of the subjective probability that a user favors them (Proposition 1), taking into various realistic preferences of the users. This means the results are shown to the users in the most appropriate way while still preserves the flexibility of the discovery solution.
- **Dominance detection** Our ranking mechanism can detect the dominance among the services easily, i.e. if any 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 (see Proposition 2).

Proposition 2. *Algorithm 2 preserves the dominance detection property in the ranking result.*

[5] http://kaon2.semanticweb.org/
[6] http://tools.deri.org/wsml2reasoner/
[7] http://lsirpeople.epfl.ch/lhvu/download/qosdisc/
[8] http://code.google.com/webtoolkit/

Table 1. Example discovery result

	UploadSpeed(q_1)	DownloadSpeed(q_2)	Price	InternetSpeed	Result
Requirements	$\geq$ 10 KB/s	$\geq$25 KB/s	$\leq$ 10 Euros	5Mbps	
Preferences	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	**rank=1**
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	**rank=2**
FilesRUDeluxe(S_3)	$\geq$ 100 KB/s	$\geq$ 500 KB/s	150CHF	5Mbps	**rejected** *(advertised price too high)*
UltraFilesPro(S_4)	$\geq$ 100 KB/s	$\geq$ 500 KB/s	100CHF	10Mbps	**rejected** *(advertised price too high, demanded Internet speed unavailable)*
WSGetNewsXignite(S_5)	–	–	100CHF	1Mbps	**rejected** *(functionality unsatisfied)*
ThemesHotel(S_6)	–	–	–	–	**rejected** *(functionality unsatisfied)*

Proof. Consider two services S_a and S_b where S_a is strictly better than S_b in term of a quality parameter q_c and better than or equal to S_2 in all other quality criteria $q_k, k \neq c$. This assumption implies that $q_{bc} \prec q_{ac}$, $\widehat{q_{bc}} \prec \widehat{q_{ac}}$, $q_{ak} \succeq q_{bk}$, and $\widehat{q_{ak}} \succeq \widehat{q_{bk}}$. One can verify that $p_{ab} > p_{ba}$ and $p_{aj} \geq p_{bj}, j \neq a, b$. Consequently, we have $P_a = (p_{ab} \prod_{j \neq a,b} P_{aj}) > (p_{ba} \prod_{j \neq a,b} P_{bj}) = P_b$. This means S_a gets a higher rank than S_b according to Algorithm 2. □

We also performed some preliminary experimental results running our semantic search engine using the 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 μ and the rank ρ 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 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, running, and comparison of the results of different variants of the reputation-based QoS estimation, matching, and ranking approaches, e.g., w.r.t different personalized preferences. This interesting question is subjective to our future work.

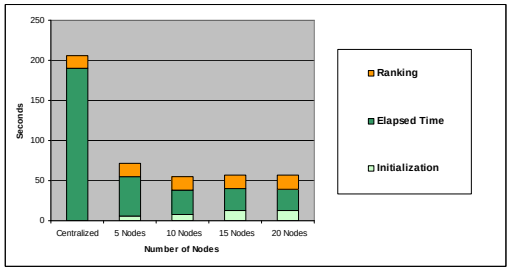

Figure 3. Experimental results

The reputation-based QoS estimation approach has also been studied under various settings, which yields very accurate and reliable results even in highly vulnerable environments. We observed the relation between the effectiveness of the reputation-based QoS estimation and other factors, such as the percentage of trusted users and reports, the rate of cheating users in the user society and the various behaviors of users. Due to limited space, we refer the reader to [44], where these experiments are presented in detail.

5 Related Work

Regarding the modeling and specification of QoS, many proposals have been devised to extend the original WSDL [17] and UDDI [6] standards to describe Web services' quality capabilities, e.g., [15, 20, 22, 29, 43]. O'Sullivan et al. [35] propose an approach to formally describe various non-functional properties (NFPs) of services (including payment, price, availability, obligations, rights, security, trust, quality, discounts, and penalties, etc.) in a

domain independent manner.

In the Semantic Web service research community, although the official OWL-S framework [3] only provides a limited way of describe service's QoS, i.e., through pairs of (key,value) of different quality attributes in the service profile, the semantic modeling of QoS has recently gained much interests. Other frameworks like SWSO [4] and WSDL-S [19] mostly focus on the functional aspects of the services and not yet provide expressive ways for describing service QoS. [47] introduces the OWL-QoS ontology for augmenting the OWL-S framework with more flexible specification of QoS parameters for Web services. [24] presents an ontology-based transformation of different notions of the WS-Agreement standard. This work on modeling of SLAs is similar to our semantic modeling of QoS requirements and offerings, yet it mostly focuses on the maintenance and monitoring of the service's SLA while we pay more attention to the issue of discovery and selection of services based on a user's QoS criteria and preferences.

Our work complements the non-functional properties specification of the WSMO framework [16], tailored to address the discovery of semantic-enabled services based on both their functionality and QoS features. Actually, at the time we start working on our conceptual model for the QoS-based service discovery, the WSMO framework has only provided limited supports for describing non-functional properties via (name, value) pairs [16]. The latest version of the WSMO framework [1] defines non-functional properties with complex constraints over their values via logical expressions, which is nearly identical to our conceptual model. However, since we have not yet had enough tools and supports from the WSMO communities, we have to base our implementation on the previous version of the WSMO model. The switching of our current implementation to the new WSMO model is straightforward.

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 cnd can be interpreted as the combination of the Agreement Context, Expiration, and Qualifying Condition in the WS-Agreement specification [15]. 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 $\langle C'(qi), cnd \rangle$ 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 have shown in the paper.

Amongst the major efforts in using QoS criteria in service discovery are those of Prof. Sheth's group, e.g., [12, 34,41]. [12] presents an approach for modeling, computing, and discovery of composite services in a workflow based on a number of important QoS criteria: time, cost, and reliability. [41] expresses business functionality and quality via WS-policy and utilizes rule-based reasoners to select appropriate providers for a certain request. This solution has certain similarity to our work, but our model is much more extensible. For example, we allow users to specify their personalized rules in finding (matching) appropriate services. Various preferences of users as well as reputation information of services are also incorporated into the selection and ranking mechanism.

[34] introduces a framework for automatically matching services and requests in terms of their agreements based on WS-Agreement standard. This work also uses multiple ontologies to express user preferences and enables the customizable matching. Our approach is comparable to this solution in terms of the expressivity of the QoS model, the flexibility of the matching and ranking algorithms. Moreover, our solution enables the inclusion of different reputation mechanisms into account for evaluating services' QoS and can be beneficial from the optimization of service queries due to our view of the discovery process as a query execution plan. Another work on Description Logic-based matchmaking of services with requests based on QoS criteria is [47]. Comparing to ours, this approach does not include a model for ranking services as well as for the personalization of the discovery process.

GlueQoS [46] only considers the syntactics matching of the quality policies of providers with the user's request which limited the expressiveness and flexibility of the QoS declaration of the provider.

The personalization of QoS-based service selection also interests various research efforts. Wagner et al [7–9, 18] introduce an interesting approach to personalized discovery of services by expanding the original queries according to various user's preferences, each time relaxing a (soft) constraint in the request to obtain more results. Our approach works in a different way by directly querying the knowledge-base of the search engine to get all relevant services, taking into account various preferences of users and produce the final personalized ranking of services. We believe that we can obtain similar results as those of [7–9, 18], while minimizing the number of calls to the reasoning engine. Actually, the performance of the discovery engine can be increased via standard optimization techniques thanks to our design of the whole discovery process as a adaptive query execution plan.

[26] presents a simple mechanism for selecting services

based on comparing of the values of their non-functional properties according to user's preferences and the recommendations of the other users. The set of preferences for users is somewhat limited and the selection model does not consider matching between possibly complex quality requirements of the users and the offers by various providers.

[30] proposes another approach for evaluating and selecting of Web services based on the difference among their QoS capability vectors. However, they only consider a limited number of QoS parameters and the proposed QoS description model is not sufficiently expressive, e.g., the notions of context and user-preferences are not included.

[33] provides a solution for matching supplies and demands based on their non-functional properties, taking the semantics of the related attributes and user's preferences into account, which is similar to our approach. However, the use of description logic in this work in some case limits the expressiveness of the QoS capability's description, e.g., it has not yet incorporated the notion of context requirements into account. Also, the evaluation of services' quality based on their reputation information is not considered.

[40] introduces an ontology-based personalization model for enabling the service provisioning based on QoS but does not propose a concrete approach to QoS-based service discovery.

Regarding the use of reputation information in the discovery process, [13] suggests the use of dedicated servers to collect the feedback of consumers and then predict future performance of published services. [11] proposes an extended implementation of the UDDI standard to store QoS data submitted by either service providers or consumers and suggest a query language (SWSQL) to manipulate, publish, rate and select those QoS data from the repository. According to [25], the reputation of a service should be computed as a function of three factors: different ratings made by users, service quality compliance and its verity, i.e., the changes of service quality conformance over time. However, these solutions have not yet addressed the trustworthiness of QoS reports produced by various users, which is important to assure the accuracy of the QoS-based selection and ranking results. [28] rates services in terms of their quality with QoS information provided by monitoring services and users. The authors also employ a simple approach of reputation management by identifying every requester to avoid report flooding. In [14], services are allowed to vote for quality and trustworthiness of each other and the service discovery engine utilizes the concept of distinct sum count in sketch theory to compute the QoS reputation for every service. However, these reputation management techniques are still simple and not robust against various cheating behaviors, e.g., collusion among liars with varying actions over time. Consequently, the quality of ranking results of those discovery systems will not be assured if there are lots of colluding dishonest users trying to boost the quality of their own services and badmouthing about others. Other work [31, 36, 37, 39, 42], which mainly uses third-party service brokers or specialized monitoring agents to collect performance of all available services in registries. We believe these approaches to be expensive to deploy in reality.

We differentiate the features of our QoS-enabled discovery solution with the most relevant related work in Table 2 via 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 results are 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 evaluation mechanisms to estimate services' quality
- OPTE: the easy parallelization and optimization of the whole discovery process

A ✓ in Table 2 denotes that the corresponding feature is supported and a * implies that the issue is (partially) addressed by some work in the mentioned group.

6 Conclusions

In summary, our approach presents an overall framework for service discovery based on both functionality and quality aspects. The presented framework is highly extensible and customizable, which adequately takes into account most important issues: semantic modeling of QoS, personalized matchmaking and ranking of services, and service's QoS reputation. The view of the whole discovery process as a query execution plan also enables the optimization of discovery queries via dynamic adaptive query execution techniques, thereby enhancing the scalability and performance of our approach.

Table 2. Comparison of our framework with others

	Ours	[34]	[41]	[47]	[26]	[7–9, 18]	[33]	[28]	[11, 13, 14, 25, 31, 36, 37, 39, 42]
QEXP	✓	✓	✓	✓			✓		*
QEXT	✓	✓	✓	✓	✓	✓	✓	✓	*
SWSE	✓	✓	✓	✓	✓	✓	✓		*
CXTE	✓	✓	✓	✓					
REPE	✓				*			✓	✓
PMCH	✓	✓				✓	✓		
PRNK	✓	✓			✓	✓	✓	✓	*
FLEX	✓								
OPTE	✓								

7 Acknowledgments

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 Líon project funded by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

References

[1] *D2v1.3. Web Service Modeling Ontology (WSMO)*. http://www.wsmo.org/TR/d2/v1.3/.

[2] *DIP Integrated project- Data, Information, and Process Integration with Semantic Web Services*. http://dip.semanticWeb.org/.

[3] *OWL-S: Web Ontology Language for Services*. http://www.w3.org/Submission/2004/07/.

[4] *Semantic Web Services Ontology (SWSO), W3C Member Submission 9 September 2005*. http://www.w3.org/Submission/SWSF-SWSO/.

[5] *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. http://www.w3.org/Submission/SWRL/.

[6] *Latest UDDI Version (3.0.2), UDDI Spec Technical Committee Draft, Dated 20041019*. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, 2004.

[7] W.-T. Balke and M. Wagner. Cooperative discovery for user-centered web service provisioning. In *ICWS*, pages 191–197, 2003.

[8] W.-T. Balke and M. Wagner. Towards personalized selection of web services. In *WWW (Alternate Paper Tracks)*, 2003.

[9] W.-T. Balke and M. Wagner. Through different eyes: assessing multiple conceptual views for querying web services. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 196–205, New York, NY, USA, 2004. ACM Press.

[10] T. Berners-Lee, J. Hendler, and O. Lassilar. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[11] A. S. Bilgin and M. P. Singh. A DAML-based repository for QoS-aware semantic web service selection. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 368, Washington, DC, USA, 2004. IEEE Computer Society.

[12] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut. Quality of service for workflows and web service processes. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(3):281–308, April 2004.

[13] Z. Chen, C. Liang-Tien, B. Silverajan, and L. Bu-Sung. UX - an architecture providing QoS-aware and federated support for UDDI. In *Proceedings of the IEEE International Conference on Web Services (ICWS'03)*, 2003.

[14] F. Emekci, O. D. Sahin, D. Agrawal, and A. E. Abbadi. A peer-to-peer framework for web service discovery with ranking. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 192, Washington, DC, USA, 2004. IEEE Computer Society.

[15] A. Andrieux et al. *Web Services Agreement Specification (WS-Agreement) Version 2005/09*. http://www.w3.org/Submission/WS-Policy/, 2005.

[16] D. Roman et al. Web Service Modeling Ontology. *Applied Ontology*, 1:77–106, 2005.

[17] E. Christensen et al. *Web Services Description Language (WSDL) version 1.1*. http://www.w3.org/TR/wsdl, 2001.

[18] M. Wagner et al. Towards Semantic-based Service Discovery on Tiny Mobile Devices.

[19] R. Akkiraju et al. *Web Service Semantics*. http://www.w3.org/Submission/WSDL-S/, 2005.

[20] S. Bajaj et al. *Web Services Policy Framework*. http://www.w3.org/Submission/WS-Policy/, 2006.

[21] A. Friesen and S. Grimm. *SWS Discovery Module Specification. DIP Project Deliverable D4.8*. http://dip.semanticweb.org/documents/D4.8Final.pdf.

[22] 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.

[23] 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.

[24] H. Jin and H. Wu. Semantic-enabled specification for Web Services agreement. *International Journal of Web Services Practices*, 1:13–20, 2005.

[25] S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(user ranking, compliance, verity). In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 200, Washington, DC, USA, 2004. IEEE Computer Society.

[26] M. Kerrigan. Web service selection mechanisms in the web service execution environment (WSMX). In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1664–1668, New York, NY, USA, 2006. ACM Press.

[27] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[28] Y. Liu, A. Ngu, and L. Zheng. QoS computation and policing in dynamic web service selection. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 66–73, New York, NY, USA, 2004. ACM Press.

[29] 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.

[30] Z. Luo, K. Qian, D. Cai, and J. S. Li. QoS driven web services assessment and selection. *Int. J. Services Operations and Informatics*, 1(1–2), 2006.

[31] E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. *SIGecom Exch.*, 3(1):24–31, 2002.

[32] F. Naumann. Data fusion and data quality. In *Seminar on New Techniques and Technologies for Statistics*, Sorrento, Italy, 1998.

[33] T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled matchmaking in an electronic marketplace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 321–330, New York, NY, USA, 2003. ACM Press.

[34] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-agreement partner selection. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2006. ACM Press.

[35] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. Formal description of non-functional service properties. Technical report, Business Process Management Group, Centre for Information Technology Innovation, Queensland University of Technology, Australia, February 2005.

[36] M. Ouzzani and A. Bouguettaya. Efficient access to Web services. *IEEE Internet Computing*, pages 34–44, March/April 2004.

[37] C. Patel, K. Supekar, and Y. Lee. A QoS oriented framework for adaptive management of web service based workflows. In *Proceeding of Database and Expert Systems 2003 Conference*, pages 826–835, 2003.

[38] 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, Trondheim, Norway, 2-3 September 2005*, 2005.

[39] S. Ran. A model for Web services discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.

[40] C. Ribeiro, N. S. Rosa, and P. R.F.Cunha. An ontological approach for personalized services. In *AINA '06:*

Proceedings of the 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06), pages 729–733, Washington, DC, USA, 2006. IEEE Computer Society.

[41] N. Sriharee, T. Senivongse, K. Verma, and A. P. Sheth. On using ws-policy, ontology, and rule reasoning to discover web services. In *INTELLCOMM*, pages 246–255, 2004.

[42] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, and J. Schiller. A concept for QoS integration in web services. In *Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops*, volume 00, pages 149–155, Italy, 2003.

[43] V. Tosic. *Service Offerings for XML Web Services and Their Management Applications*. PhD thesis, Department of Systems and Computer Engineering, Carleton University, Canada, 2004.

[44] L-.H. Vu, M. Hauswirth, and K. Aberer. QoS-based service selection and ranking with trust and reputation management. In *Proceedings of the OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005 Proceedings, Part I*, volume 3760, pages 446–483. Springer-Verlag GmbH, 2005.

[45] 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.

[46] E. Wohlstadter, S. Tai, T. Mikalsen, I. Rouvellou, and P. Devanbu. Glueqos: Middleware to sweeten quality-of-service policy interactions. *icse*, 0:189–199, 2004.

[47] C. Zhou, L.-T. Chia, and B.-S. Lee. Web services discovery with daml-qos ontology. *International Journal of Web Services Research(JWSR)*, 2(2):44–67, 2005.