# Through Different Eyes – Assessing Multiple Conceptual Views for Querying Web Services

Wolf-Tilo Balke
Computer Science Department
University of California
Berkeley, CA, USA
balke@eecs.berkeley.edu

Matthias Wagner
Future Networking Lab
DoCoMo Communications Laboratories Europe
Munich, Germany
wagner@docomolab-euro.com

## ABSTRACT
We present enhancements for UDDI / DAML-S registries allowing cooperative discovery and selection of Web services with a focus on personalization. To find the most useful service in each instance of a request, not only explicit parameters of the request have to be matched against the service offers. Also user preferences or implicit assumptions of a user with respect to common knowledge in a certain domain have to be considered to improve the quality of service provisioning. In the area of Web services the notion of service ontologies together with cooperative answering techniques can take a lot of this responsibility. However, without quality assessments for the relaxation of service requests and queries a personalized service discovery and selection is virtually impossible. This paper focuses on assessing the semantic meaning of query relaxation plans over multiple conceptual views of the service ontology, each one representing a soft query constraint of the user request. Our focus is on the question what constitutes a minimum amount of necessary relaxation to answer each individual request in a cooperative manner. Incorporating such assessments as early as possible we propose to integrate ontology-based discovery directly into UDDI directories or query facilities in service provisioning portals. Using the quality assessments presented here, this integration promises to propel today's Web services towards an intuitive user-centered service provisioning.

## Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services– *Web-based services.*

## General Terms
Algorithms, Human Factors, Design.

## Keywords
Web services, Semantic Web, personalization, cooperative service discovery, user profiling, preference-based service provisioning.

## 1. INTRODUCTION
Web services are expected to provide an open platform not only for electronic B2B interaction, but also for the provisioning of so-called user-centered services, i.e. B2C services that can provide useful information and a variety of service offers to support users in a modern mobile lifestyle. Though the capabilities of such services are still relatively simple, their sophistication will grow with the improvement of (wireless) networks, bandwidths, and client device capabilities. However, finding the adequate service for subsequent use of each individual user becomes a more and more demanding problem. Given the convergence of networks in forthcoming (mobile) environments and the evolving innovative business models for third party service deployment (e.g. NTT DoCoMo's i-mode service certification/licensing model for mobile service portals [19]) the variety of services is even expected to grow. Making an informed choice of the 'right' service will therefore include matching individual users' preferences or dislikes against the concepts and capabilities of the services offered.

Usually the interaction process for Web services consists of three distinct phases: a discovery of possible services, the selection of the most useful, and the subsequent execution. In understanding what a service actually offers the first two phases are crucial and the general acceptance of user-centered services will depend on the solutions of still demanding problems in interaction like cooperative querying. As shown in [4] and [5] the discovery and selection processes of user-centered Web services involves a high degree of respect for user preferences to be flexible enough for real world use. In that respect providing user-centered services strongly differs from the well-defined capabilities of traditional B2B services. As a running example of a typical user-centered service we will use an extension of the cooperative restaurant booking Web service presented in [4]: restaurant booking services subscribe to the least general applicable node along a complex service ontology for a number of characteristics. A service request can then be performed including a choice of various individual categories. However, the individual services offered will usually only more or less match all the user's expectations. Ranking services with respect to requests is thus an ongoing challenge, as is also evident from the research areas of IR or Web search engines for information provisioning.

Service providers almost always can anticipate some typical interactions with their services. For our example typical tasks are for instance booking a certain restaurant for a specific evening, finding a suitable restaurant in the vicinity for lunch, etc. The characteristics and input parameters for Web services for restaurant booking thus usually contain a number of general input values that can be specified in a service request/query: the name of the restaurant, its location, its specific address, the type of cuisine, the date and time for a booking, its price range or even third party content like recommendations (e.g. the Zagat reviews). However, from a service provisioning point of view the nature of these parameters strongly differs. A user expecting to book a certain restaurant on a specific evening will expect that the request may be

Copyright is held by the author/owner(s).
*WWW 2004*, May 17-22, 2004, New York, NY USA.
ACM 1-58113-912-8/04/0005.

granted or may fail depending on current reservations of that restaurant for the given date, but relaxing the constraints of the date given or booking a different restaurant for the evening might simply not do. In contrast a user simply wishing for a close-by restaurant to have lunch will rarely provide such fixed terms as a restaurants name, but rather use descriptive terms like a preferred cuisine and an approximate location.

Distinguishing such query stereotypes like 'book a table at the 'Chez Panisse' for the 12/3/03 8:00 pm' and 'give me the name and address of a Chinese restaurant in the commercial district of San Francisco with medium price range' and the subsequent personalization of service provisioning also needs different types of input parameters. Whereas simple variables like the restaurant's name or a certain category in a clear request can be handled in an exact match fashion, more fuzzy attributes in a somewhat tentative request like an approximate location or the choice of cuisine have to be understood as a user's preferences with respect to certain concepts (soft constraints). In the area of the Semantic Web the management of such concepts is usually done by the very powerful tool of ontologies that describe a generalization hierarchy of such concepts. In the course of this paper we will show how to open up service provisioning to the better understanding, adequate handling and quality assessment of each individual user's intentions and preferences. The contribution of the paper thus is twofold:

- On one hand we relate the use of ontologies and the handling of conceptual views like given by the Semantic Web to cooperatively evaluating preferences for each specific user
- On the other hand we show how to effectively deal with the problem of relaxing multiple conceptual views for more complex queries and give quality measures to assess the most useful results for each specific user.

Both contributions can be expected to improve the service provisioning of user-centered Web services and help to boost their usability and thus subsequently their acceptance.

## 2. SEMANTIC REGISTRY ENHANCEMENTS

Today Web services are usually provided via an Internet wide network of services registries given by the Universal Description Discovery and Integration (UDDI) [21]. UDDI builds on the Web Service Definition Language WSDL [7] which features basic information about providers of a service and technical service invocation details. Even though UDDI has become the de facto standard in the field it suffers from a major shortcoming: the information offered on individual services is rather limited. A yellow-page-style lookup mechanism provides the service interface together with a short verbal description of what task the service performs. Mainly targeted a human Web service experts and developers, advanced query capabilities and cooperative matchmaking, however, are still lacking.

Research in the area of the Semantic Web seeks a solution to this unsatisfying situation, e.g. [20][4]. Generally speaking, the Semantic Web fosters a population of the Web with content and services having formal semantics and rich service descriptions. Several semantic frameworks for Web services are currently emerging with DAML-S [2] and W3C's recently established OWL-S [9], [10] initiative as the most prominent approaches. We have built our previous work on DAML-S as a relatively mature ontology-based approach to the description of Web services that tries to provide a common ontology of services for the Semantic Web. Building on top of DAML+OIL [8] the Web service representations in DAML-S consist of a service profile for advertising and discovering services, a process model giving a detailed description of a service's operation and a service grounding providing details on how to interoperate with services via message exchange.

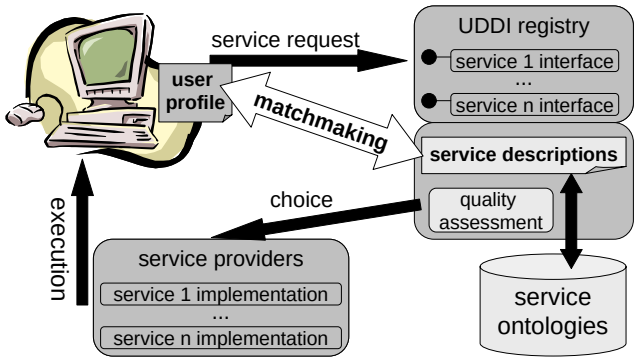


**Figure 1: Concept of enhanced UDDI service registries**

Figure 1 shows a schematic view of our semantically enriched Web service provisioning concept. A user states personal needs and preferences in an enhanced service request. The enhanced UDDI registry matches this request against the descriptions of all registered services. The actual matching can be carried out using cooperative database technology like shown in [4]. The query is split in hard and soft constraints where the hard constraints are processed as filter conditions, whereas the soft conditions can be relaxed if necessary. If no user-specific preferences are given with the service request, the relaxation follows the domain-specific conceptual views of the service ontology given by the service providers or portal operators. To distinguish between several possible relaxations the quality assessment, which is the main aspect of this paper, will evaluate the degree of match for each service with respect to the original user query and offer all best matches. After a certain implementation has been chosen, the service provider will execute the service and deliver the result.

From a service provisioning viewpoint centralized and publicly available service ontologies can be understood as a default service conceptualization or the most common service concept hierarchy, i.e. encoding common and widely accepted knowledge or world/domain knowledge. Due to the hierarchical nature of ontologies a user asking for a specific service will also be served with any more specific service concept subsumed by his request. On the other hand, in the case where a best match to his initial request is not available he/she might also be satisfied with more general services from a super-class of the requested one. This is determined by a relaxation step in the service ontology, i.e. a generalization of concepts along the lines of the ontology.

## 3. ONTOLOGY-BASED WEB SERVICE DISCOVERY AND SELECTION

In this section we provide a brief overview of the use of ontologies for relaxation of soft query constraints and show how common knowledge serves as a default for cooperative retrieval.

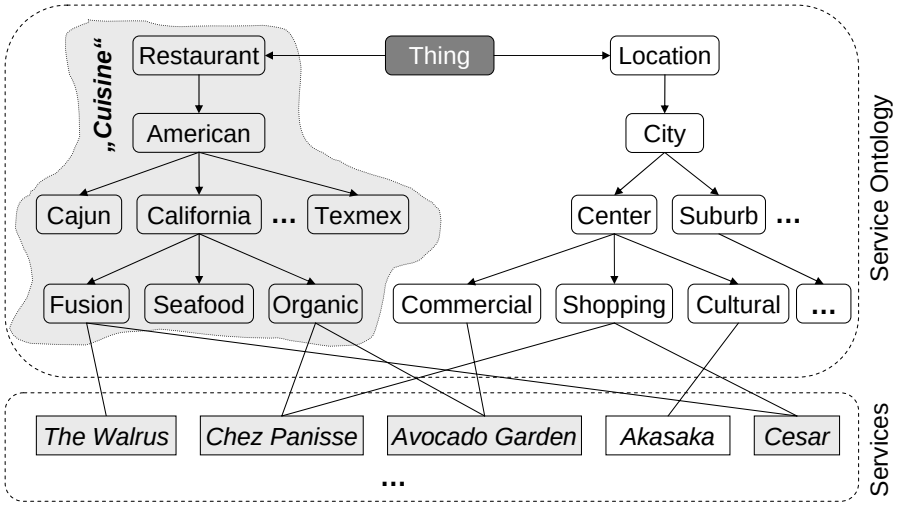


**Figure 2: Service Ontology for restaurant booking.**

## 3.1 Service Ontologies, User Preferences and Usage Patterns

The purpose of a service ontology is to describe the kinds of entities available in a service repository and how they are related. To this end service ontologies may include descriptions of service classes, properties and their instances (the actual services that are eventually selected for execution). A basic service ontology is depicted in Figure 2. Here restaurant booking services are classified according to their cuisine and their location in a city. For instance the restaurant 'Chez Panisse' serves 'Organic' food with 'Organic' being a specialization of the 'Californian' cuisine (as well as of 'American'). Furthermore the restaurant 'Chez Panisse' is located in the 'Shopping' district which itself is part of the city 'Center'. We have used W3C's Web Ontology Language (OWL) and its predecessor DAML+OIL to enrich DAML-S service profiles in Web service repositories [4][5]. Modeled in OWL the most general 'Restaurant' and 'Location' concepts are anchored in 'owl:Thing' – the most common concept of any ontology.

A restaurant booking service using the service ontology from Figure 2 might on one hand assume that a user asking for 'a restaurant featuring American cuisine' will be well served by all restaurants with e.g. Cajun, Californian or Texmex cuisine, since they are all instantiations of American cuisines. On the other hand, if a user asks for 'a Californian fusion cuisine restaurant' and no such service should be registered, implicitly relaxing the query to all Californian restaurants and offering restaurants with organic cuisine or Californian seafood to our user will be more helpful than just stating the empty result. And even if a user has different conceptions (e.g. cuisines being related based on their flavors) or explicit preferences (a Chinese restaurant rather than an Italian one), an ontology-based discovery/selection model is still useful. [4] shows in detail how to deal with these cases by overwriting the default ontology with an explicitly provided (or implicitily derived) generalization hierarchy of a user somewhat similar to the view definitions proposed by [14]. Since for our assessment framework here the exact kind and classes/values of an ontology do matter less than its actual structure in each instance, such overwritings by user specified conceptions are always possible to facilitate.

We advocate the use of service ontologies together with a proprietary notion of basic user preferences and typical service usage patterns for the stepwise refinement of service requests in a cooperative service provisioning environment. While the basic approach and combination of ontologies, preferences and patterns is published elsewhere [5] we will now concentrate on enhancements to the relaxation along the lines of ontologies alone: unlike the specialization of a request, a generalization can result in severe changes of the initial query semantics. This is especially true, if several relaxation steps have to be performed until a match can be found. At the point of relaxing a constraint to the root of an ontology, the respective constraint can even be considered as entirely dropped. But nevertheless, since an ontology resembles common knowledge (and thus implicit preferences), for high quality service provisioning offering somewhat related features is usually still a better default than just returning an empty result set.

## 3.2 Multiple Conceptual Views

Individual users might have quite specific ideas about differing domain concepts (conceptions) or very clear expectations how to be served differing from the usual domain assumptions (explicit preferences), but also implicit preferences play an important part. Consider for instance location-based services, e.g. for restaurant booking. If a user asks to book 'a Chinese restaurant for dinner', the common domain knowledge tells us that this restaurant should be in the vicinity (e.g. a 30 miles area) of his current or usual whereabouts and we can add this information as an implicit constraint for better provisioning quality. A user in San Francisco would usually be annoyed by offers of Chinese restaurants in Hong Kong no matter how good their actual quality or rating is. If this general assumption would not hold, however, (e.g. if a user wants to fly to Hong Kong in the morning and then have dinner there) he or she would have stated this unusual detail already within the query and had asked for a 'Chinese restaurant in Hong Kong for dinner'. Such explicit information within a service request is provided due to the psychological notion that though users want a service to know what is sensible (like they expect to be served in human-human interaction), no user expects a service to be clairvoyant. Thus, not only having further (explicit) knowledge of a user, but also assuming typical behavior, concept hier-

archies given by ontologies can be used as good default relaxation hierarchies for user preferences. Should, however, some preferences or a specific conception be given, the underlying ontology has to be exchanged against the user-provided terms or concepts.

We introduce the notion of conceptual views on a service ontology to account for all the different interests a user wants to express in a service request. Such a conceptual view is modeled as a clipping from the full service ontology that starts with the most general concept associable with a specific user interest. For this paper we will for the ease of understanding assume conceptual views to be non-overlapping, tree-shaped clippings from the full service ontology where each service is registered with the node that describes its value with respect to the most specific characteristics. An example conceptual view is indicated as a grey shaping in figure 2: the view named 'Cuisine' is basically a sub-ontology only concerned with the classification of restaurants according to the offered type of food. Whereas the restaurants 'The Walrus', 'Chez Panisse', 'Avocado Garden' and Cesar are classified as being 'Fusion' or 'Organic' places the restaurant 'Akasaka' is not reachable in this view as it is only classified as located in the cultural district. As we will discuss in the remainder of this paper multiple conceptual views can be used to account for different interests a user wants to express in a service request and the relative importance between them. In the case of the restaurant booking example it is conceivable that a user values the fulfillment of location constraints over cuisine constraints if he/she is only up for a quick work lunch. However, for the ambitious 'hobby gourmet' this might be just the other way round on the weekend. Thus we will need ways to assess the respective quality of different relaxation schemes to allow users to make an informed choice.

# 4. RELAXING MULTIPLE ONTOLOGIES

Let us now focus on problems that arise when multiple soft constraints have to be relaxed over the service ontology. We will first look at our sample scenario, investigate the relaxation of conceptual views and then discuss some quality considerations.

## 4.1 Relaxation Plans for Multiple Selection Predicates

Let us consider our restaurant booking service from above. A typical query would be "Find a Californian fusion cuisine restaurant in the commercial district of Berkeley". Here we will have to deal with two soft constraints: the type of cuisine and the location. Assuming that we do not have more specific information about the user's preferences both constraints could be relaxed along the two default conceptual views of the service ontology of figure 2.

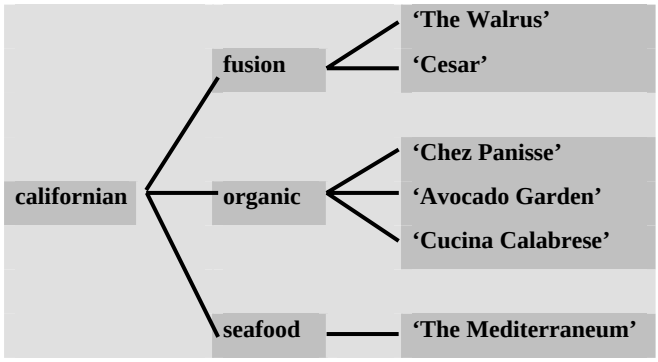


**Figure 3: The cuisine ontology with respective instances**

Figures 3 and 4 show the full first two concept levels of the respective conceptual views with some instances. So in figure 3 we can for instance see that there is a service for a restaurant called 'The Walrus' which is classified as offering fusion cuisine and as such, also offering Californian cuisine. The relaxation of query predicates over such a view is straightforward. If the query predicate specifies 'fusion cuisine' restaurants, we would have the choice between the respective instances, here 'The Walrus' and 'Cesar'. If for some reason there would be no fusion cuisine restaurants registered, or the instances cannot satisfy some other constraints (like booking for a certain date), we will relax along the ontology to the more general concept of Californian cuisine and can also consider the restaurants that are registered under the 'organic' and 'seafood' characterizations of California cuisine.

The technical problem of how to relax single conceptual views with adequate query languages over cooperative database systems is in detail addressed e.g. in [4] and [5]. The beauty of the design is that all details of a UDDI or DAML-S style description for each service together with some more characteristics (e.g. taken from RDF statements of a restaurants homepage) can be stored in a classic relational database by the service provider and can be searched using a declarative query language extended by preference constructors like shown in e.g. [13]. Thus an added-value service using semantically meaningful content can be provided quite easily.

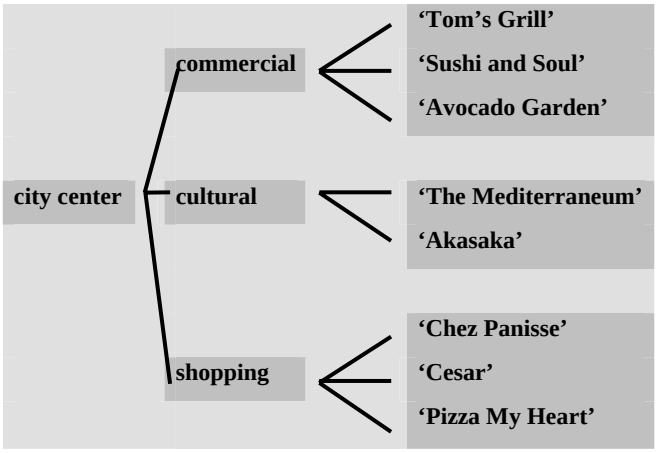


**Figure 4: The location ontology with respective instances**

A more serious problem arises when several soft query constraints over various conceptual views have to be evaluated. Consider for instance the query on a 'fusion cuisine restaurant in the commercial district'. We can easily verify that in our example even though 'The Walrus' and 'Cesar' are fusion cuisine, they are not registered in the commercial district. Likewise 'Tom's Grill', 'Sushi and Soul' and 'Avocado Garden' are in the commercial district, but they do not offer fusion cuisine. Aiming at a cooperative retrieval behavior we are left with three choices: relaxing the constraint on the cuisine, relaxing the constraint on the location or relaxing both constraints. When relaxing the cuisine to 'californian' we would retrieve the service of 'Avocado Garden' the only Californian cuisine restaurant within the commercial district. Relaxing the location-based ontology would result only in the service of 'Cesar' the only fusion cuisine restaurant in the city center. Finally relaxing both constraints would result in all Californian cuisine restaurant within the city center, i.e. 'Avocado Garden', 'The Mediterraneum', 'Chez Panisse' and 'Cesar'. Thus

different kinds of relaxation will usually result in essentially differing answer sets. This problem will remain if more relaxation steps have to be taken. If we have relaxed only one constraint we could e.g. decide to relax this property even further or relax another property of our service during the next step. Let us first define the task of finding the 'best' service under relaxation

**The Problem of Best-Matching Service Provisioning:**
Given various characteristics that describe all services in a service ontology, a concept hierarchy (conceptual view) for each characteristic and a user request stating a number of hard and soft constraints, the best-matching service is given by all services that:
- fulfill all the hard constraints
- fulfill all soft constraints with a minimum amount of relaxation of characteristics with respect to a suitable quality measure

So given that each service is registered in one concept node for each characteristic given by the respective conceptual view, and all services not fulfilling the hard constraints have been filtered, the problem of selection over multiple constraints comes down to deciding what is 'a *minimum* amount of relaxation'. Obviously the basic task is finding a service that is registered with *all* concepts (or any of their respective sub-concepts) as specified by the query, a 'perfect match'. But if there is no such service registered, the decision what soft constraints to relax and how far they are relaxed, is paramount for the quality of provisioning.

## 4.2 Basic Service Quality Considerations

Since the decision about the relaxation scheme is important for the output, some way of considering which scheme to follow is needed. Usually ontologies are of a qualitative nature. A superclass / subclass taxonomy is established, but there is no knowledge of the 'degree' or the relative distance between different concepts. However, such knowledge could crucially change the utility of certain relaxation plans. Relaxing more refined ontologies or views will generally hurt the user preferences less than relaxing already coarse views or ontologies. The less general the concept, the more refined are the sets of objects that will be offered to the user, and flooding the user with too much too general content is avoided. Let us first take a closer look at merely qualitative views on ontologies of comparable granularity, etc. (i.e. relaxing one constraint is introducing the same amount of generalization as relaxing any other) and then investigate ways to deal with quantitative measures in the following section.

For scenarios of merely qualitative preferences and their relaxation for the restricted class of ceteris paribus preferences [16] proposes a scheme of ordering different objects in the result set according to the count of necessary relaxation steps from the top or the bottom of the hierarchy or simply the relative distance to all violated query constraints. Since we assumed symmetrical views on ontologies, also in our relaxation problem a similar concept will help us to understand what should be relaxed preferably and what this means for the objects in the result set. Let us first show the approach of simply counting the relaxation steps from the violated constraints. We will label the services we found in each step by the number of necessary steps to find them. But first we need to define the necessary concept of relaxation paths.

Given tree-shaped conceptual views of a service ontology that arranges concepts or values with respect to certain service characteristics using a generalization semantics, a **relaxation path** is a path along the edges of each conceptual view that leads from a base concept to the respective root of the view. Usually this base concept is specified in a service request or user query and relaxing along the relaxation path leads to an increasing generalization of this concept. Assuming that all services have been assigned to the node of their first appearance along the relaxation path (i.e. they are registered to any of the respective node's sub-trees of concepts, but not to an earlier node of the relaxation path) we get a chain of concepts with all services registered under the aspect of least necessary level of generalization.

An example for a relaxation path can be easily derived from figures 2 and 4. If a user is primarily interested in the commercial district, the appropriate nodes of the relaxation path would be 'commercial district', 'city center', 'city' and 'location'. The services registered in the nodes are e.g. 'Tom's Grill', 'Sushi and Soul' and 'Avocado Garden' for 'commercial district'. The node 'city center' would also contain all services registered to its subconcepts, (i.e. 'The Mediterraneum', 'Akasaka', 'Chez Panisse', 'Cesar' and 'Pizza My Heart') and so on. Figure 5 shows the first two steps of respective relaxation paths for both conceptual views in figures 3 and 4 focusing only on the services registered in both views. As we pointed out we will always assume tree-shape views for the course of this paper. Please note that all the concepts easily can be transferred to the case where sub-concepts can have multiple parent nodes. In this case the node along the relaxation path would consist of the intersection of the different parent concepts, or the intersection of their registered services respectively. This generalization has already been successfully employed in a similar fashion for mapping queries between differing ontologies by [18]. Distances then can simply be measured by the minimum distance, if multiple paths for relaxation should be available.

Let us now see, how relaxation can be done using an unlabeled relaxation graph (see figures 3 and 4). If we begin by either relaxing the cuisine or the location constraint of our query we would have to assign a quality value of 1 to both the 'Avocado Garden' and 'Cesar'. Thus in terms of quality they are incomparable, which closely resembles our missing knowledge of what kind of relaxation the individual user would prefer. If there should be more ways to relax constraints to encounter a service, we will always count the minimum number of relaxation steps necessary. Relaxing both constraints again leads to quality values of 1 for 'Avocado Garden' and 'Cesar' and a value of 2 for 'The Mediterraneum' and 'Chez Panisse', because they have only been seen by relaxing both constraints and thus are probably less desirable for the user. However, relaxing the cuisine ontology two steps (i.e. to all American cuisines) and sticking to the commercial center constraint might result in 'Tom's Grill' turning up also with a value of 2. This is because in the naïve model the semantic difference between 'deep' relaxation and 'broad' relaxation is considered the same. To be sensitive to the differing implications of broad and deep relaxation with respect to generality we will use our concept of relaxation paths and show how to use labels along this path to get to a more sophisticated relaxation paradigm.

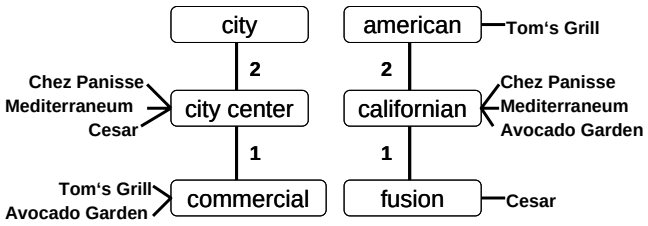


**Figure 5: Relaxation paths and assigned services**

Usually the generalization throughout ontologies will become quickly rather unspecific with decreasing distance to the root. Hence a broad relaxation strategy (breadth first relaxation) often is preferred to deep relaxation steps. Summing up the distances like before, but weighing each relaxation step with the relative distance to the original query term can implement this. Consider our query for 'fusion cuisine' restaurants in the 'commercial district'. So for example the 'Avocado Garden' and 'Cesar' need each only one relaxation step with a distance of one to the original constraint (cf. labels in figure 5), so their quality value is 1. 'Chez Panisse' and 'The Mediterraneum' both need two relaxation steps with a distance of 1 each resulting in a value of 2. In contrast 'Tom's Grill' also needs only two relaxation steps, but whereas the first step has a distance of 1, the second step already shows a distance of 2. Thus the final value for 'Tom's Grill' is 3 (i.e. 1+2*1) and we can now effectively distinguish between deep and broad relaxation. Depending on the nature and granularities of the ontology or views we can of course also use higher weightings for the deep relaxations, for instance $10^{(\text{distance}-1)}$. So the first step will be weighted by 1, the second deep step by 10, the third deep step by 100, and so on. Since the broad relaxation steps are still simply added up, this will 'punish' deep relaxation and avoid too broad generalizations of constraints. If we always want to punish deep steps symmetrically until all constraints in turn are relaxed at least to the same level a factor of $(\text{number\_of\_ontologies})^{(\text{distance}-1)}$ will be adequate as shown in the following lemma.

**Lemma 1: Weightings to Foster Broad Ontology Relaxation** Given $n$ soft query constraints with their respective relaxation hierarchies. To always prefer a broad relaxation scheme, label each object by summing up the numbers of edges relaxed to find this object in each hierarchy and weigh every edge by $n^{(d-1)}$ using the number of soft constraints $n$ and the relative distance $d$ to the original query constraint.

**Proof:** Since within each depth all weightings are the same, it is obvious that within a certain depth of the hierarchy any object seen with less relaxation steps has a smaller label than an object that needs more steps. Thus if we e.g. have to relax two constraints within a level this object will always be labeled with a higher weight than an object that needed only one relaxation independently of which constraints have been relaxed.

We still have to show that if we do a step with a deeper distance, an object $O$ encountered there always gets a higher label than any object $P$ encountered in all hierarchies only with relaxations up to a lower distance. We will do that by showing that the minimum label for object $O$ is higher than the maximum label for object $P$. Let us assume that in order to encounter object $O$ we have to relax at least one constraint to a distance of $k$. The minimum label for $O$ thus is given by relaxing only a single constraint to distance $k$ and not having to relax any other constraint. Hence object $O$'s label is given by $(n^0+n^1+\ldots+n^{(k-2)}+n^{(k-1)})$. The maximum possible label for object $P$ on the other hand is given by having to relax every of the $n$ constraints ($k$-1)-times, i.e. the maximum distance smaller than $k$. Thus the maximum label for $P$ is $n*(n^0+n^1+\ldots+n^{(k-2)}) = (n^1+\ldots+n^{(k-2)}+n^{(k-1)})$ and thus $P$'s label is -even relaxing all constraints to a maximum- at least by 1 smaller than the best possible label for $O$. ■

Thus in relaxing constraints for equally important conceptual views an adequate algorithm would be the processing of decreasing levels of quality, i.e. finding services with increasing weightings. Starting with the minimum possible relaxation the algorithm will always work over an entire sequence of services with the same quality index and return all the discovered services of the lowest level found, together with their quality estimation. This is important for having to restart the algorithm at the previous point of termination, if the services discovered so far should not have been sufficient and the evaluation of lesser quality levels becomes necessary. Similarly, knowing the labeling technique and the views involved a user can also specify a maximum quality value up to which he/she is willing to accept more general services. For our algorithm we will assume a declarative query mechanism on UDDI directories enhanced by all the feature characteristics described by their respective conceptual views like presented in [4].

However, symmetrically relaxing just the same number of steps even when assuming views of the same granularity and importance with respect to the user query, will generally not lead to our desired broad relaxation scheme with as little generalization as possible. Imagine a query that specifies two soft predicates of which one is a leaf node of a view, whereas the other predicate specifies a direct descendant node of the root in the respective view. If no perfect match should be given, our strategy would result in relaxing either of the two constraints a single step. But whereas the relaxation by a single step from our leaf node usually leads to a slight generalization allowing a few more services for selection, the relaxation step in our second constraint would relax to the root node and thus offer the total number of services registered in the entire ontology for the respective characteristic, i.e. entirely drop the second constraint. Obviously that would not be a sensible behavior. So we do not only have to punish deep relaxation steps but even more severely refrain from relaxations the closer we are to the root.

This concept will be implemented in our relaxation algorithm by letting the longest possible relaxation path determine the weightings for all constraints (again assuming a comparable level of detail throughout our ontology). The weightings for edges along the relaxation path will then be assigned in each conceptual view in descending order starting from the root down to the concept specified by the query. Thus the relaxation of all concepts at least to the same level of generalization is enforced before having to relax already more general concepts. In our example from above the relaxation path for a leaf node concept would be assigned weightings as given by the height of the conceptual view, whereas the relaxation path for a concept node right below the root would be assigned the highest possible weighting. Thus (following lemma 1), our leaf node concept will have been relaxed to a generalization level of the respective concept right below the root before the second constraint is relaxed for the first time. Now we are ready to present an algorithm for relaxation of symmetrical constraints incorporation a breadth first paradigm and a minimum level of generalization strategy.

**Algorithm: Symmetrical Constraints Breadth First Paradigm**

1. Pose the query containing only the hard constraints against the enhanced UDDI / DAML-S directory.

    1.1. If an empty result should be returned, terminate the algorithm outputting the empty result set.

    1.2. Repose the query with all soft constraints included.

    1.3. If a non-empty result should be returned for the expanded query, terminate the algorithm and output the respective services as perfect matches with a relaxation level of 0.

2. Given *n* the number of soft constraints we have to label each edge along the relaxation path from the concept specified by the query to the root node in every conceptual view (cf. lemma 1).

   2.1. Among the *n* views find the longest possible relaxation path and set *maxdepth* as the maximum depth of all ontologies relative to the class specified in the service request

   2.2. For every view label the relaxation path starting from the root by $n^d$ down to the concept specified by the query (d := (*maxdepth* –1) to 0 descending)

3. For i = 1 to $\Sigma\, n^j$ ($1 \le j \le$ *maxdepth*)

   3.1. Start with the query including all hard and soft constraints and build statements containing any possible relaxation with a weighting of i, i.e. relax in turn all conceptual views by one step up to the point of reaching the desired weighting. Due to construction of the weighting this will result in a breadth first strategy.

   3.2. If any of the statements produced in 3.1. retrieves a non-empty result set, collect results of all possibilities and terminate the algorithm.

In this algorithm step 3 can efficiently be implemented using an A*-Algorithm that successively explores the differently weighted tree edges finding all possible combinations for each quality weighting. However, please note that not every weighting is possible to reach by relaxing constraints.

We will now exemplify the above algorithm through different examples: consider the three conceptual views X, Y and Z given in figure 6 and assume a user has posed a query requesting the characteristics X3, Y2 and Z6 as soft constraints. Let us assume all hard filter conditions have already been satisfied, but the basic query for our soft constraints fails, i.e. no service with these capabilities is registered. Step two of our algorithm will now label the relaxation paths like shown in figure 6 (bold edges and shaded vertexes). Starting with all services having quality values of one, concept Z6 is generalized to Z3 and the query is reposed with constraints X3, Y2 and Z3. If we still should have no matching services we have to go on relaxing. A query for a value of two is not possible, but for a value of 3 we can relax X3 to X2 and repose the query with constraints X2, Y2 and Z3. Let us assume we still have not found a result the next quality value would be 4. For this we have two possibilities and have to unite the results of the query on X2, Y2, Z3 and the query X3, Y2, Z2. The next possible quality value would be 7 with a query on X2, Y2, Z2. Please note that we indeed have relaxed all constraints to same level until we relax any constraint to the top level (e.g. in the view Y) for the first time. The algorithm would terminate at the latest after relaxing all views to the top level with a quality value of 34.

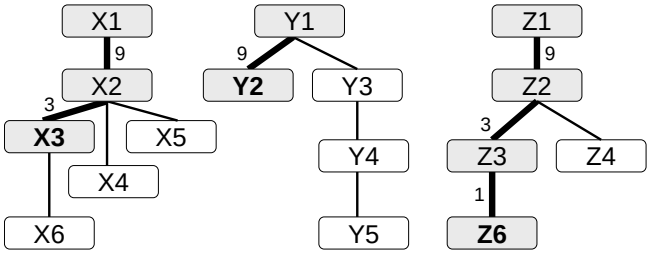


**Figure 6: Conceptual views with labeled relaxation paths**

## 4.3 Quantitative Service Quality Measures

In the last section we have seen an effective scheme for the case of symmetrical relaxations under the assumption of equal usefulness, i.e. one broad step was as useful as any other broad step of a comparable level. But in real world applications query constraints are not always only of a qualitative, incomparable nature. Deeper knowledge of individual user's preferences or knowledge of stereotypical usage can become interesting parameters in assessing the quality of different relaxation schemes. Tuning the factor to a certain ratio for each application (x broad steps equal 1 deep step) will express the desired semantics in each instance. The exact coefficient used for the discrimination of deep relaxation in each instance, however, will typically strongly depend on the domain, the total number of soft query constraints and the respective granularity of the views / ontology (i.e. the semantic level of detail) used. Views that are modeled with a very fine granularity can be relaxed introducing a smaller degree of generalization to the service request results than would be introduced by those views that are modeled rather coarsely anyway. Hence, a deep relaxation step in a very detailed ontology might be worth only three or four broad relaxations of other user constraints, whereas a deep step in a coarser ontology used within the same query might add up to the worth of ten broad relaxation steps or even several deep relaxation steps of other constraints. Likewise very flat hierarchies with many subclasses to each node are not suited too well for deep relaxation. So the discrimination will in each application depend on:

- The relative semantic importance of a view with respect to the user request
- The maximum depth of each conceptual view,
- Its total number of (sub-) concepts and
- The (average) number of instances in each concept.

The relative semantic importance of a view can usually only be determined by directly consulting the user. But in the following we will give an overview of techniques that will generally help to deal with problems of relaxing views with different granularities. As a rule of thumb we can state that the relaxation of views having a low maximum depth and rather high numbers of services attached to each node should be delayed as long as possible. Our technique of starting the view graph labeling from the root node already helps facilitating this rule. If a shallow conceptual view is used (usually an indication for coarse modeling) together with a more detailed view, even the edges to leaf nodes in the shallow ontology will be assigned rather high weightings unlike the leaf nodes in ontologies with a rather high depth. This behavior is, however, not always the best choice. If unlike in figure 6 the respective depths of conceptual views differ by a considerable amount, we should not simply delay the relaxation of shallow views, but have to insert several intermediate steps in the more detailed views before relaxing the next step in a coarse view. Figure 7 shows pairs of views X, Y and X', Y'. In both cases the depth of X, X' is only two whereas the depth of Y, Y' is four. That means that in terms of relaxation we can assume that for some reason the more shallow ontologies X and X' are modeled rather coarsely. On the left hand side in figure 7 we can see the labeling scheme from our algorithm. Ontology Y would be relaxed to Y3 or even Y2 before a single step in X would be relaxed. On the right hand side we can see a better labeling scheme with interleaved relaxation steps (two steps in Y' for a step in X').

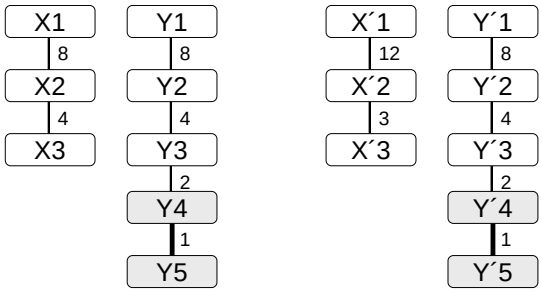


**Figure 7: Differently labeled relaxation paths**

Since the interleaving of relaxation steps with respect to the maximum depth generally seems a fairer approach, we will incorporate this behavior into our algorithm. If the maximum depth of some conceptual views should severely differ, we will assume a coarser level of detail and find out how many steps in the most detailed view represent a single step in the coarser view. We then re-label the coarse view beginning from the root by adding so many of the appropriate sequence of weightings, as steps in the detailed view are necessary. For instance in figure 7 we can easily see that a single step in X (depth 2) represents two steps in Y (depth 4) and thus we would have to re-label the first edge by the sum of the two highest weights of Y (8+4) and its second edge by the sum of the next two weights (2+1). Following this scheme we can gain the more suitable relaxation weights of view X'. In terms of our quality assessment algorithm that means that we have to reconsider the labeling of the relaxation paths in step two and replace the respective section by the following:

> 2.1. Among the *n* conceptual views find the longest possible relaxation path multiplying possible steps in each view relative to the class specified in the service request by the view's respective factor of *q*, where *q* is given as the integer part of the result of dividing the maximum view depth by the maximum depth of the current view (i.e. *q* steps in the most detailed view represent one step in this view). Set the maximum value for *maxdepth*.
>
> 2.2. For every conceptual view label the relaxation path starting from the root by $n^d + n^{d-1} + \ldots + n^{d-q+1}$ (i.e. the sum of the first *q* weights in terms of the most detailed relaxation path) down to $n^d + n^{d-1} + \ldots + n^0$ with d:= (*maxdepth* –1).

A second possibility to control the relaxation properties of multiple conceptual views is the incorporation of user preferences giving a preferred relaxation order. Incorporating such preferences into the weights along the ontology, however, is a very difficult problem in its own rights and therefore beyond the scope of this paper. For the case that a simple ordering of relaxation for the views is given by the user, we can use double-labeled relaxation paths like for the tree patterns in [1]. The second label for each node is e.g. the respective rank of the view in a specified relaxation ordering or the number of the node's respective sub-concepts. In the case that for the execution of step 3 in our algorithm more than one query should be possible, the relaxations can then be executed minimizing the sum of second labels and retrieval can be terminated whenever a result occurs. Also for the case that a prioritization of relaxations is given (cf. [13]), the respective relaxation scheme is straightforward: The prioritized view is successively relaxed until a first result set is retrieved. Then the second, third, etc. views are used to break ties. However, when it comes to integrating weights from preferences into relaxation weights deeper research is still needed.

# 5. RELATED WORK

While in the above discussions we assumed the existence of different conceptual views to a given ontology, the actual creation of these views is beyond the scope of this paper. Multiple views as abstractions of data sources are a well understand concept in classical databases systems. Yet the concept of such views has only recently been addressed in the context of the Semantic Web through the proposal of the view definition language RVL for the low level ontology language RDFS [14]. RVL uses a declarative query language for the creation of virtual schemas which in turn serve as views on existing complex ontologies. Please note that although we merely focused on tree-shaped clippings from OWL ontologies as simple relaxation hierarchies in our examples, the presented concepts are general enough to be used with only the slightest adaptations together with other types of ontologies and more complex views, e.g. virtual RVL views.

Choosing the 'right' Web service for execution has been considered in several ways. For legal or economical points of view especially assurance structures guaranteeing that a service performs the desired task like [11] or [17] have been addressed. However, when negotiating about execution guarantees or costs the semantic content of a service has to be understood and its specific capabilities have already to be agreed on. Taking a more user-centered view the notion of services' reputation for subsequent selection [15] or the quality assessment for the negotiation of service level agreements [3] have been proposed. However, these approaches focus on conceptual designs omitting algorithms how to assess the quality in each instance. The most complete framework with respect to heterogeneous environments featuring multiple ontologies is given by [18] where the notion of information loss for query reformulation is defined. Unlike our work presented here, where multiple conceptual views of a service ontology occur in a single request, this work, however, deals with the loss of information when a query has to be translated from one into another ontology (e.g. in order to pose it to a different data source). Thus it is rather concerned with the problem of ontology mappings.

The area of service request relaxation over ontologies also shows some similarities with database query relaxation frameworks like given in [6] or [13] and especially recent work on querying semi-structured data like in XML databases. In the case of XML the DTD of a document defines its structure together with the (semantic) type of data within each node. The main focus of querying in that area is on building queries without perfect knowledge of documents structure or the exact data it contains. Exploiting the set of labels given by a XML document's DTD as ontology in term of the documents' structure [12] uses the result sets of queries to define the semantic equivalence of alternative query expressions, however without relaxing concepts within queries. The area of relaxation for tree-shaped queries not only on a structural level ('relax to any descendant node instead of child node'), but also on a limited semantic level ('find author of document instead of book') is in detail addressed in [1]. Here also weightings along the edges of trees comparable to our user preference-driven quality assessments in section 4.3 are discussed. Our work differs mainly in that we can rely on fine-granular concept ontologies that are custom made by domain experts and used in central service provisioning portals or UDDI / DAML-S directories. Not only are we able to exploit semantics by a far larger extent than previous work, but we also provide means to derive sensible weightings for edges within conceptual views based on general user preferences and a fair relaxation paradigm.

The general area of enhancement of UDDI goes back to describing the capabilities of Web services on a more detailed level by using ontology languages like DAML+OIL [8]. An example for the efficient mapping of DAML+OIL capability descriptions onto UDDI records is given in [20]. Our approach for result quality assessment here is facilitated by the database-based approach for UDDI enhancement featured in [4]. Using cooperative database technology and an extended declarative query language like the one given in [13] this framework features a good implementation framework for our quality assessment. Queries using hard and soft constraints can be automatically rewritten using the relaxed concepts and posed against a database of service descriptions using suitable relaxation ontologies. However, in terms of quality these languages feature only a qualitative result set under the notion of Pareto-optimality. Quantitative quality constraints that limit the flood of incomparable results delivered by the exponential growth of Pareto sets with the number of soft constraints in the request are not considered. A first study of such quality measures is given in [16] for the restricted class of ceteris paribus preferences like discussed in section 4.2.

## 6. SUMMARY AND OUTLOOK

In this paper we presented a framework for the discovery and selection of Web Services based on personalized quality assessments for individual service users. Starting with a set of conceptual views over a service ontology that express a generalization hierarchy of concepts (conceptual views) for different services' capabilities and characteristics, we proposed to enhance today's UDDI / DAML-S registries by a matching component that will not only perform a filtering of services according to user specified terms, but also allows for cooperative matchmaking between service descriptions and the individual user's preferences. Focusing on the quality assessment component of such an enhancement we described in detail how to deal with different kinds and multiple instances of conceptual views. The views in our framework contain the domain-specific understanding of concepts or the common knowledge that users typically will expect when trying to find an adequate service for execution. If no service that offers all required capabilities can be found, relaxing along these views will step by step generalize the services' requested features until a best possible match can be found. Thus cooperative behavior can be introduced for improved service provisioning.

We focused on controlling these relaxation steps implementing a breadth first strategy control flow to delay far-reaching generalizations for as long as possible. We also discussed the influence of relaxation plans for various views, which may differ in their granularity or accuracy of discrimination and the influence of individual user preferences for relaxation orders. For the case of views with differing level of details we gave an adequate scheme to balance the control flow. Nevertheless, the exact instantiation of the views and the adequate weightings chosen still will usually differ between application areas. Anticipating stereotype interaction patterns or experiences of past interactions, however, the service provider usually is able to also provide some suitable default ontologies for cooperative matchmaking within UDDI / DAML-S registries or managed service portals. In any case the provisioning of suitable facilities for the assessment of Web service quality can be expected to become a central part of service provisioning and will essentially influence the future acceptance of Web service offers by individual clients.

In this paper we have restricted conceptual views to tree-shaped clippings of ontologies with view elements being exclusively related through 'is-a' relationships (stating an explicit generalization of concepts in a superclass/subclass fashion). Of course also other relationships within ontologies might be available for relaxation tasks in service requests; on the other hand their semantic meaning will usually be somewhat more difficult. Since complex ontologies commonly contain named relationships between entities that might be used to make views more flexible and query relaxation more meaningful, an important future work item will be to break down this restriction on relationships of the 'is-a' type. With our algorithms' focus on relaxation paths as an abstraction of the underlying views, our general framework for quality assessment can be expected to be extensible also to these new types of views in a straightforward manner independently of the exact type of view the relaxation path was derived from. If a relaxation of a constraint along a certain relationship is backed by sensible relaxation semantics (i.e. is meaningful), however, has to be checked in each individual instance.

Furthermore, our future work will focus on a tighter integration of individual user preferences into the quality assessment process like addressed in section 4.3. Choosing the adequate weightings does not have an obvious semantics. The meaning of 'relaxing one constraint is *two-times* better than relaxing another constraint' can only be guessed, what about *three-times*, etc.? The area between quantitative quality assessments like e.g. re-weighting techniques or relevance feedback as known from the area of IR, and the purely qualitative approaches like Pareto optimality of solutions like given in [13] offers a vast variety of possibilities to explore for real world applications. Also here the notion of stereotypical usage of services and the grouping of users with similar intensions might lead to improved service provisioning. We believe that using and extending our framework is a vital step towards getting a better understanding of these topics. In any case assessing quality of service request results in a semantically sensible way promises to pave the road to cooperative provisioning for user-centered services.

## 7. ACKNOWLEDGMENTS

We would like to thank Achim Leubner and Anthony Tarlano for helpful comments and suggestions. This work was partially funded by an Emmy-Noether-Grant of the German Research Foundation (DFG).

## 8. REFERENCES

[1] S. Amer-Yahia, S. Cho, D. Srivastava. Tree Pattern Relaxation. In *Proc. of the Int. Conf. on Extending Database Technology (EDBT'02)*, Prague, Czech Republic, 2002.

[2] A. Ankolenkar, M. Burstein, J. Hobbs, et. al. DAML-S: Web Service Description for the Semantic Web. In *Proc. of the Int. Semantic Web Conf. (ISWC'02)*, Sardinia, Italy, LNCS 2342, Springer, 2002.

[3] W.-T. Balke, A. Badii. Assessing Web Services Quality for Call-by-Call Outsourcing. In *Proc. of the Int Workshop on Web Services Quality (WQW'03)*, Rome, Italy, 2003.

[4] W.-T. Balke, M. Wagner. Cooperative Discovery for User-centered Web Service Provisioning. In *Proceedings of the First International Conference on Web Services (ICWS'03)*, Las Vegas, USA, 2003.

[5] W.-T. Balke, M. Wagner. Towards Personalized Selection of Web Services. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003) Alternate Track on Web Services*, Budapest, Hungary, 2003.

[6] S. Chaudhuri. Generalization and a Framework for Query Modification. In *Proc. of the Int. Conf. on Data Engineering (ICDE'90)*, Los Angeles, USA, 1990.

[7] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.

[8] D. Connolly et al. DAML+OIL Reference Description. *W3C Note*, December 2001.

[9] DAML. OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.html#foot29

[10] DAML. OWL-S 1.0 Release. http://www.daml.org/services/owl-s/1.0/

[11] M. Jakobsson, M. Yung. On Assurance Structures for WWW Commerce. In *Proc. of Int. Conf. on Financial Cryptography (FC'98)*, Springer LNCS 1465, Anguilla, British West Indies, 1998

[12] Y. Kanza, Y. Sagiv. Flexible Queries over Semistructured Data. In *Proc. of the ACM Symp. on Principles of Database Systems (PODS'02)*, Santa Barbara, USA, 2001.

[13] W. Kießling, G. Köstler. Preference SQL - Design, Implementation, Experiences. In *Proc. of the Int. Conf. on Very Large Databases (VLDB'02)*, Hong Kong, China, 2002.

[14] A. Magkanaraki, V. Tannen, V. Christophides, D. Plexousakis. Viewing the Semantic Web Through RVL Lenses. In *Proc. of the Int. Semantic Web Conf. (ISWC'03)*, LNCS 2870, Sanibel Island, USA, 2003.

[15] E. M. Maximilien, M.Singh. Conceptual Model of Web Service Reputation. In *SIGMOD Records* 31(4), 2002.

[16] M. McGeachie, J. Doyle. Efficient Utility Functions for Ceteris Paribus Preferences. In *Proc. of Conf. on Artificial Intelligence and Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI'02)*, Edmonton, Canada, 2002.

[17] G. Medvinsky, C. Lai, B. Neuman. Endorsements, Licensing, and Insurance for Distributed System Services. In *Proc. of the ACM Conf. on Computer and Communications Security*, Fairfax, USA, 1994

[18] E. Mena, V. Kashyap, A. Illarramendi, A. Sheth. Imprecise Answers in Distributed Environments: Estimation of Information Loss for Multi-Ontology based Query Processing. In *International Journal of Cooperative Information Systems (IJCIS)*, 9 (4), 2000.

[19] NTT DoCoMo home page. http://www.nttdocomo.com/home.html, 2003.

[20] M. Paolucci, T. Kawamura, T. Payne, K. Sycara. Importing the Semantic Web in UDDI. In *Proc. of the Int. Workshop on Web Services, e-Business and the Semantic Web (WES'02)*, Toronto, Canada, 2002

[21] UDDI. The UDDI Technical White Paper. http://www.uddi.org.