Quantified Matchmaking of Heterogeneous Services

Michael Pantazoglou, Aphrodite Tsalgatidou, and George Athanasopoulos

Department of Informatics & Telecommunications,
National & Kapodistrian University of Athens, 15784, Greece
{michaelp, atsalga, gathanas}@di.uoa.gr

Abstract. As the service-oriented computing paradigm and its related technologies mature, it is expected that electronic services will continue to grow in numbers. In such a setting, the course of service discovery could yield many alternative yet heterogeneous services which, by all means, may be of different type and moreover distinguished by their quality characteristics. To come through such situations and ease the task of service selection, service search engines need to be powered by an efficient matchmaking mechanism, which will abstract requesters from service heterogeneity and provide them with the means for choosing the service that best fits their requirements, among a wide set of services with similar functionally. In this paper, we present an efficient service matchmaking algorithm, which facilitates the task of heterogeneous service selection, whilst combining and exploiting the syntactic, semantic, and Quality-of-Service (QoS) properties contained in service advertisements.

Keywords: service discovery, service matchmaking, service ranking, heterogeneous services.

1 Introduction

Service discovery plays a fundamental role in service-oriented development, allowing developers to find and re-use existing units of software for rapidly building distributed applications. According to advocates of the *Service-Oriented Architecture* (SOA), electronic services will continue to grow in numbers as the related technologies mature. Accordingly, a query for a specific type of functionality could yield many alternative services, which may be of different type (e.g. web or peer-to-peer services) and moreover distinguished by different Quality-of-Service (QoS) characteristics. This case is better illustrated through the description of the following real-world scenario.

The IT department of a private clinic has decided to develop a service-oriented application, which will enable direct interactions between doctors, patients, as well as other partners (e.g. insurance companies, pharmacy companies, external doctors, etc). Fig. 1 depicts an excerpt of this application, where the doctor asks for a second opinion, based on anonymous medical information retrieved from the clinic's internal database. The clinic has already established partnerships with a number of external doctors as well as with other clinics and hospitals, which offer the appropriate services for the establishment of this type of communication. These services are potentially of different type, they may have been described with the use of different

K. Aberer et al. (Eds.): WISE 2006, LNCS 4255, pp. 144 – 155, 2006.

description protocols, and they may also have different quality-of-service characteristics. For example, hospitals may have exposed this type of functionality through a web service interface, while the external doctors may be contacted and asked for a second opinion through a specialized p2p service running on their PDAs or mobile phones. Consequently, the developer faces the problem of having to select among all these similar services the one that is most suitable for the specific task.

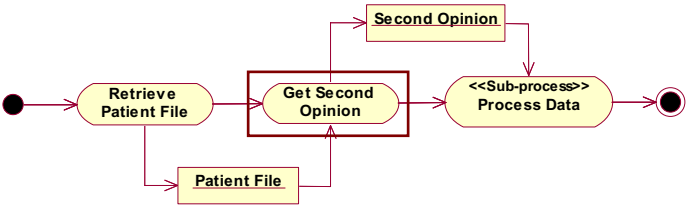

Fig. 1. A sample healthcare application requiring the use of a "get second opinion" service

To overcome such an arduous task, the course of service discovery needs to be leveraged by an efficient matchmaking mechanism, which will abstract requesters from the technical complexity and heterogeneity of the various service advertisements and provide them with a means for making the right selection. Naturally, the result of such a mechanism will be a list of appropriately ranked, similar services.

In this paper, we present a matchmaking algorithm, which aims at facilitating the task of service selection among a set of alternative services, by quantifying their similarity. Among the strong points of the proposed matchmaker are: 1) its ability to deal with the underlying heterogeneity of existing services, both in terms of their type and their description protocols, 2) the fact that it produces results by combining syntactic, semantic, and QoS service characteristics. The matchmaking algorithm has been fully implemented by the specification of the *Unified Service Query Language*, namely *USQL*, which is described in [1]. The detailed description of the language goes beyond the scope of this paper; however, USQL will be used as a means to express requirements, in order to validate the proposed algorithm.

Briefly, the rest of the paper is structured as follows: Section 2 introduces the basic concepts that form the basis for the definition of the matchmaking algorithm, which is described in Section 3. In Section 4, a preliminary experiment is conducted, with the use of USQL, so as to give some early results regarding the precision of the algorithm. Section 5 compares our work with related efforts and emphasizes its main contributions. Finally, Section 6 concludes with a small discussion on future work.

2 Basic Concepts

In this paragraph, we briefly present the conceptual model that has driven the definition of the matchmaking algorithm. Described by the W3C Body, the *Service-Oriented Model (SOM)* [3] defines that a service groups the message interactions it can be engaged in through an interface. Moving along this high-level assumption, we established in previous work a *Generic Service Model (GeSMO)* [2]. GeSMO moves

one step further by defining message interactions as operations, while also declaring that a service may expose multiple interfaces, which in turn offer one or more such operations. The following conceptual model (Fig. 2) depicts our perception on the notion of service and its fundamental parts.

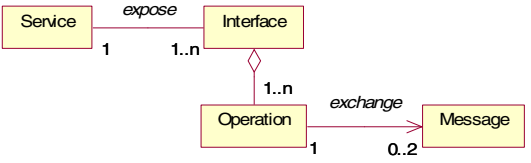

Fig. 2. High-level conceptual model for services

All concepts in the above model are given by default a syntactic description, but it is also possible that semantics and/or QoS properties are assigned to some of them. In our approach, we assume that services, service operations as well as their exchanged messages can be semantically described. In addition, service operations can be given QoS properties, such as availability, reliability, processing time, etc (Fig. 3). Consequently, when querying for a specific type of functionality in a service-oriented setting, it becomes natural that requirements can be expressed at the service level, the operation level, as well as the message level. Moreover, a service query may consist of syntactic, semantic, and QoS requirements.

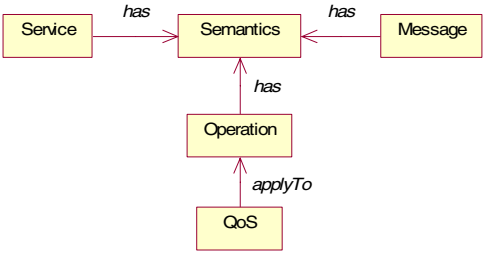

Fig. 3. Semantics and QoS as they are supported by our conceptual model

Having described our conceptual view on services, we proceed in the following with the definition of our matchmaking algorithm.

3 The Matchmaking Algorithm

The process of service matchmaking can be basically described as follows: *Given a service request comprising a set of requirements, check against a set of service advertisements and calculate the degree of match for each corresponding service.* The degree of match quantifies the suitability of each service with respect to the original request and it usually takes the form of a percentage amount.

A service request contains a series of requirements, which can be either simple (e.g. the name of the service provider, the type of input/output parameters, the processing time of a service operation, etc.), or complex. As the term implies, complex requirements consist of other requirements, which in turn may be either simple or complex. For example, a series of operation-level requirements, such as the operation name and semantics, the operation inputs and outputs, operation QoS, etc. are grouped as one complex requirement regarding the whole operation. In the next paragraphs, we define a set of formulas which are used to calculate the degree of match for simple and complex requirements.

3.1 Matching Simple Requirements

According to the conceptual study that was described in the previous section, the requester may express syntactic, semantic, and/or QoS requirements upon formulation of a service discovery request. To process such requests, a matchmaking algorithm should be primarily equipped with the appropriate mechanisms for the calculation of the degree of match of simple syntactic, semantic, and QoS requirements.

Similarity distance measure has been established as one of the most popular matchmaking techniques and is being used extensively in the areas of data mining and web information retrieval [16], [15]. The relative theory is basically evolved around the notion of *distance functions*, a definition of which can be found in [4]. Our proposed matchmaking algorithm utilizes this technique in order to calculate the degree of match of such requirements in a service discovery request.

Definition 1. Given a simple requirement q and the value a of the corresponding property in a service advertisement, their degree of match d is defined as follows:

$$d = 1 - dist(q, a) \ . \tag{1}$$

As we can see, the calculation of the degree of match employs *dist*, a normalized similarity distance measure function. More specifically, *dist* will return 1 in cases where q and a do not match at all, 0 in cases where q and a perfectly match, and an appropriate value between 0 and 1, if where there is a partial match between q and a. In other words, the more a matches q, the more *dist* leans towards 0 and, reversely, the less a matches q, the more *dist* leans towards 1. Consequently, according to equation (1), the degree of match d equals to 1, if the advertised property exactly matches the requested one, while it takes the value of 0, if there is no match.

3.1.1 Syntactic Requirements

The expression of a syntactic requirement implicitly states a set of accepted values for the corresponding advertised property. Consider an example, where the requester is looking for services being offered by Microsoft. The implied set of accepted values for the service provider property comprises any phrase or word containing the keyword *'Microsoft'*. The example reveals that, the result of applying syntactic matchmaking has a binary nature, that is, either the advertised property's value belongs to the set of accepted values implied by the syntactic requirement, or it does not. The following definition captures this assumption:

Definition 2. Given a simple syntactic requirement q_{syn}, its implied set of accepted values S_q, and the value a_{syn} of the corresponding syntactic property in a service advertisement, their similarity distance measure function is defined as:

$$dist_{syn}(q_{syn}, a_{syn}) = \begin{cases} 0 & , a_{syn} \in S_q \\ 1 & , a_{syn} \notin S_q \end{cases}. \quad (2)$$

Going back to the above example, $dist_{syn}(q_{syn}$, *'Microsoft Corporation'*$) = 0$, while $dist_{syn}(q_{syn}$, *'IBM'*$) = 1$. Respectively, the degree of match in the first case would be equal to 1 (i.e. the simple syntactic requirement was fully met) while in the second case it would be equal to 0 (i.e. the simple syntactic requirement wasn't met).

3.1.2 Semantic Requirements

In measuring the distance measure between two semantic concepts we take into consideration their hierarchical relation in the ontology graph. Particularly, the semantic relations *exact*, *plug-in*, *subsume*, and *fail*, between two semantic concepts, which have been described in [5], can be quantified with the definition of the following semantic distance measure function:

Definition 3. Given a simple semantic requirement q_{sem} and the value a_{sem} of the corresponding semantic property in a service advertisement, their distance is measured as follows:

$$dist_{sem}(q_{sem}, a_{sem}) = \begin{cases} 0 & , \text{if } q_{sem} \text{ and } a_{sem} \textit{ exactly} \text{ match} \\ 1/3 & , \text{if } q_{sem} \text{ is a } \textit{plug-in} \text{ of } a_{sem} \\ 2/3 & , \text{if } q_{sem} \textit{ subsumes } a_{sem} \\ 1 & , \text{if } q_{sem} \text{ and } a_{sem} \text{ don't have an immediate relation } (\textit{fail}) \end{cases} \quad (3)$$

The values assigned to each of the identified semantic relations reflect their semantic order, as it has been explained in [5].

3.1.3 QoS Requirements

As in the case of syntactic search criteria, a QoS requirement defines a set of accepted values for a given QoS property, from the requester's point of view. For example, if the requester requires that the operation must be at least 99.9% available, given the fact that the operation availability cannot be validated more than 100%, the implied set of accepted values for this specific QoS property will be [0.999, 1], where percentages have been expressed as real numbers between 0 and 1.

Due to the fact that most QoS properties are numerical, we can refine the definition of the similarity distance measure function so that it provides us with more realistic results. Such definition is given as follows:

Definition 4. Given a simple QoS requirement q_{qos}, its implied set of accepted values S_q, and the value a_{qos} of the corresponding QoS property in a service advertisement, their distance measure, $dist_{qos}$, is defined as:

$$dist_{qos}(q_{qos}, a_{qos}) = \begin{cases} 0 & , a_{qos} \in S_q \\ \dfrac{|q_{qos} - a_{qos}|}{\max(q_{qos}, a_{qos})} & , a_{qos} \notin S_q \end{cases} \quad (4)$$

Note that, the above function is applicable only to QoS properties which take the form of positive numbers. Such properties include the operation availability, reliability, processing time etc. Another appropriate similarity distance measure function should be defined for non-numerical QoS properties (e.g. security).

3.2 Matching Complex Requirements

While the matchmaking process of simple requirements is straightforward and depends primarily on their type (i.e. whether these refer to syntactic, semantic, or QoS properties of the service), matching complex requirements requires an advanced formula which must be able to also satisfy a number of conditions imposed by intuition and practice.

Upon formulation of a service request, some requirements are often given higher priority than others. Prioritized requirements are particularly useful when the task of service discovery is performed at design-time, where the ultimate selection of a service is human-driven. In such cases, requesters may assign different priorities to their needs, so as to distinguish the real important requirements from the less important or optional ones. For instance, during the development of a service-oriented application, the satisfaction of the requested capability of an operation (i.e. semantics of the operation) would be considered more important than matching its actual signature, since the developer has the ability to adapt his/her application accordingly, for the operation to fit in. It is therefore imperative for a matchmaking algorithm to take into consideration the likely different priorities of requirements in a service request. Intuitively, the existence of requirements with different priority levels should have an impact on the calculation of the degree of match: the higher the priority level of a requirement is, the more the requirement's degree of match should affect the calculation of the total degree of match. Moreover, it makes sense to claim that, if the most important requirements in a query are not satisfied, the resulting degree of match should be leaning towards zero.

In the definitions that follow, we capture the essence of the aforementioned conditions in order to render the matchmaker of complex requirements intuitive.

Definition 5. Let $P = \{p_1, p_2, ..., p_k\}$ be an ordered set of k priority levels p_i, $1 \le i \le k$, where $p_1 \prec p_2 \prec ... \prec p_k$ and $Q = \{q_1, q_2, ..., q_n\}$ be an unordered set of n requirements, which have been assigned different priority levels, $k \le n$. Then, Q can be expressed as

$$Q = Q_1 \cup Q_2 \cup ... \cup Q_k . \quad (5)$$

In equation (5), Q_i, $i = 1..k$ are the un-ordered sets of n_i requirements with p_i priority level, where $Q_i \cap Q_j = \emptyset \quad , \forall i, j \in [1,k], i \ne j$, $n_i > 0$ and $\sum_{i=1}^{k} n_i = n$.

It should be noted that, the actual type and values of the priority levels do not affect the matchmaker definition. In our approach, we are only interested in the defined order between the different priority levels, which we use to group requirements according to equation (5). In this way, the matchmaking algorithm is rendered independent from the actual matchmaker implementation. For instance, in the USQL language we have specified two values for the priority level of the requirements, namely *'low'* and *'high'* [1].

In what follows, we define the formula for the calculation of the degree of match for complex requirements.

Definition 6. Let Q be an un-ordered set of n requirements, $n > 0$, with k different priority levels (see Definition 5) and *adv* be an advertisement which Q is checked against. Then, the degree of match d_m for Q is calculated with the use of the following formula:

$$d_m = \frac{1}{\sum_{i=1}^{k} n_i \cdot i} \cdot \left[k \cdot D_k + \sum_{i=1}^{k-1} \left(\frac{i \cdot D_i}{k-i+1} \cdot \left(1 + \sum_{j=i+1}^{k} \frac{D_j}{n_j} \right) \right) \right]. \tag{6}$$

In the above definition, we have $D_i = \sum_{j=1}^{n_i} d_j$, where $n_i > 0$ is the number of requirements having i priority level and d_j is the degree of match of the j^{th} requirement in subset Q_i. It can be proven that $0 \leq d_m \leq 1$, however, due to lack of space we omit the proof in this paper.

A closer look at equation (6) reveals that the defined formula abides by the intuitive conditions which were set previously. Indeed, if all requirements with the highest priority are not met (i.e. $D_k = 0$), then the resulting degree of match for Q diminishes towards zero. Also, the significance of the degrees of match of lower-prioritized requirements in calculating the overall matching degree is affected the degrees of match of the higher-prioritized ones. In other words, the higher the priority of a requirement is, the more its degree of match determines the overall calculation.

3.3 Matchmaking Against Multiple Advertisements

Up to now, we have provided definitions for the calculation of the matching degrees of simple and complex requirements against a given advertisement. Since a service request is basically a set of requirements, it can also be considered as a complex requirement and, thereby, equation (6) is used for the calculation of the overall degree of match of a service. Nevertheless, in practice, service requests are commonly checked against a number of service advertisements. To accommodate this fact, we need to reshape equation (6) with the use of matrices. First, we provide a generic definition of what a matchmaker is:

Definition 7. Let $Q = \{q_1, q_2, ..., q_n\}$ be an unordered set of n requirements, and $A = \{a_1, a_2, ..., a_m\}$ be an unordered set of m advertisements. We define a matchmaker M as

$$M : Q \times A \mapsto D. \tag{7}$$

where $D = [d_{j1}]$, $0 \le d_{j1} \le 1$, $j = 1..m$ is a $m \times 1$ matrix containing the resulting degrees of match of the advertisements.

In general, when n_k requirements with the same priority level k are checked against m advertisements, we can make use of an $m \times n_k$ matrix $R_k = [d_{ij}]$, $i=1..m$, $j=1..n$, for displaying the resulting degrees of match. Each element d_{ij} in this matrix is the resulting degree of match of the jth requirement, checked against the ith advertisement, and has been produced with the use of the appropriate formula, from the ones defined in equations (2), (3), (4) and (6). Then, the summaries of the degrees of match of all n_k requirements per advertisement ($\sum_{x=1}^{n_k} d_x$) are collectively calculated as follows:

$$S_k = R_k \cdot U_{nk} . \tag{8}$$

where U_{nk} is a $n_k \times 1$ matrix, whose elements are all equal to 1, and S_{nk} is the resulting $m \times 1$ matrix.

Definition 8. Let Q be an un-ordered set of n requirements, $n > 0$, with k different priority levels (see Definition 5). Given the equations (6), (7) and (8), the degrees of match for a number of m advertisements contained in an unordered set $A = \{a_1, a_2, ..., a_m\}$ are calculated as

$$M(Q,A) = D_m = \frac{1}{\sum_{i=1}^{k} n_i \cdot i} \cdot \left[k \cdot S_k + \sum_{i=1}^{k-1} \left(\frac{i \cdot S_i + DS_i \cdot \sum_{j=i+1}^{k} \frac{S_j}{n_j}}{k-i+1} \right) \right]. \tag{9}$$

where:

- D_m is the resulting $m \times 1$ matrix containing the matching degrees of the m service advertisements
- DS_x , $x=1..k$, is an $m \times m$ diagonal matrix, immediately produced by S_x as follows:

$$S_x = \begin{bmatrix} d_1 \\ d_2 \\ . \\ d_m \end{bmatrix} \Rightarrow DS_x = \begin{bmatrix} d_1 & 0 & ... & 0 \\ 0 & d_2 & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & d_m \end{bmatrix}. \tag{10}$$

This concludes the definition of the matchmaker.

4 Evaluation

The experiment described in this section provides some preliminary results regarding the precision of the proposed matchmaking algorithm. Going back to the scenario

described in Section 1, we will apply matchmaking against a number of alternative services for the *"get second opinion"* task (see Fig. 1). For the purposes of this early experiment, we implemented two web services (ws1 and ws2) and a JXTA p2p service (ps3) with similar functionality, yet distinguished by certain syntactic, semantic and QoS characteristics. Both web services have been described with the use of WSDL-S (see http://www.w3.org/Submission/WSDL-S/), regarding their interface and semantics, and WS-QoS (see http://www.wsqos.net), with respect to their QoS offers. For the p2p service, we used WSDL and OWL-S (see http://www.w3.org/Submission/OWL-S/) to describe it syntactically and semantically. Finally, the same ontology that was used for semantically annotating all services was also employed in the formulation of our query. The following table summarizes the three services and their properties:

Table 1. Services used for the evaluation of the matchmaking algorithm

Service	*Provider*	*Domain*	*Description*	*Operation*
ws1	Medisystem	Healthcare	%second opinion%	Capability: GetSecondOpinion Input: EpisodeId:string Output: Diagnosis:string Availability: 99.95%
ws2	eHealth	PrivateClinicServices	%second opinion%	Capability: GetSecondOpinion Input: EpisodeFile:file Output: SecondOpinion:file Availability: 99.96%
ps3	Medisystem	CardiologyServices	%second opinion%	Capability: GetSecondOpinion Input: EpisodeDesc:string Output: Diagnosis:string Availability: 99.899%

Precision. We used USQL to formulate a query for a *"get second opinion"* service in the domain of *Cardiology*, which is a sub-domain of *Healthcare*, provided by *Medisystem*, offering an operation with capability semantically described as *"GetSecondOpinion"*. The requested operation should accept an *episode file* as input, and return the *diagnosis* as output. Moreover, its availability should be equal or greater than *99.9%*. From the above requirements, the operation and domain were given *high* priority. The USQL request was produced according to these requirements and can be found in http://cgi.di.uoa.gr/~michaelp/usql-request-wise06.usql.

The application of the matchmaker formula, defined in equation (9), along with the use of formulas (2), (3), (4), and (6), produced the following degrees of match:

Table 2. Results of the matchmaking algorithm

Service	*Degree of Match*
ws1	77.42 %
ws2	41.14 %
ps3	91.31 %

According to the results, the p2p service (ps3) was found to be closer to our requirements. However, this should not come as a surprise: a closer look at Table 1 shows that, only the operation input and availability were not fully compliant with the requested ones.

5 Comparison with Related Work

A large number of research efforts have been conducted over the years, dealing with the problem of similarity measuring. Many of these approaches were adapted to service matchmaking, with special attention paid at semantic similarity between ontology classes and/or their properties [6], [7]. In [8], a novel service retrieval approach was proposed, that captures service semantics via process models, and applies a pattern-matching algorithm to locate desired services. An ontological approach was proposed in [11] which, like our matchmaking algorithm, also considers user assigned priorities in matchmaking. Following a different direction, a set of algorithms were developed in [9] which enable searching for web service operations that are similar to a given one. The underlying idea of their search engine is the grouping of inputs and outputs into semantically meaningful concepts. Thus, syntactic information in service advertisements attains semantics and can be exploited in a more fruitful manner. Many proposals have also emerged to deal with QoS matchmaking. In [10], a matchmaking framework was described, which maps QoS requirements of consumers with the published QoS information of providers, also accommodating QoS-Constraints.

As opposed to the above mentioned approaches, which are restricted to performing matchmaking against either syntactic, semantic, or QoS search criteria, our matchmaker is capable of blending all these types of criteria in calculating the service degree of match, thus supporting the specification of syntactic, semantic, as well as QoS requirements within a service query. The calculation of the degree of match of a service remains independent of the way in which the degree of match of each one of the constituent requirements is calculated. Thus, the matchmaker can be extended with as many types of requirements and their related matchmaking mechanisms as needed. In this regard, existing advanced matchmaking mechanisms like the ones mentioned above can be seamlessly embedded in our approach. The LARKS framework [12] also combines syntactic and semantic matchmaking, yet its main drawback lies in that it supports a rather static service description schema. Our matchmaker overcomes this shortcoming as it abides by a high-level conceptual model which complies with most service-oriented technologies and standards, such as WSDL, WSDL-S, OWL-S, WS-QoS, etc. Due to the abstraction of the underlying model, it is expected that the matchmaking algorithm can be applied to a wide range of heterogeneous service-oriented environments, ranging from UDDI and ebXML registry lookups to service discovery within p2p networks and/or grid virtual organizations. Other important features of our proposed matchmaking algorithm are its intuitiveness and scalability provided by its own definition (see equation (9)). Hence, it becomes possible to apply the matchmaking algorithm to any number of service advertisements in a parallel manner.

The matchmaking algorithm presented here has been fully implemented by the specification of USQL [1] and a USQL-enacting service search engine prototype [13]. The matchmaker, the USQL language, and its associated search engine have been developed in the context of the SODIUM project [14] and form part of its provided platform [13]. Within the context of SODIUM, the presented matchmaking algorithm has been applied in a number of use cases accruing from applications in the domains of *Healthcare* and *Crisis Management*, where the ultimate selection of the most appropriate service for a given task was significantly facilitated.

6 Discussion

In this paper, we presented a service matchmaking algorithm capable of assessing complex service requests against a number of service advertisements and ranking the returned results. Furthermore, we evaluated the matchmaking algorithm's precision with the conduction of a preliminary experiment. Among the matchmaker's novelties is its applicability to any type of services, provided that their descriptions are compliant with the matchmaker's underlying service model. Also, as opposed to most of the related work that we have looked at, our matchmaking algorithm considers syntactic, semantic, and quality requirements in calculating the overall matching degree of a service. All in all, we believe that, the main contribution of our approach is the provision of an intuitive, unified matchmaking approach, in terms of service and requirements heterogeneity, which is capable of dealing with complex service requests. Our matchmaker alleviates users from the cumbersome task of separately matching syntactic, semantic, and QoS requirements and manually combining the results.

In the future, we aim at selectively accommodating in our matchmaking algorithm the most promising among the aforementioned efforts, by exploiting the fact that its definition remains independent from the way individual degrees of match are produced. This will allow us to conduct more extended experiments, which we expect to give interesting results. Finally, our immediate plans include extending the USQL language to support more than two priority levels for service requirements, so that we have the opportunity to better evaluate the matchmaking algorithm.

Acknowledgement. This work is partially supported by the Special Account of Research Funds of the National and Kapodistrian University of Athens under contract 70/4/5829 and by the European Commission under contract IST-FP6-004559 for the SODIUM project [14].

References

1. Tsalgatidou, A., Pantazoglou, M., Athanasopoulos, G. (2006) *Specification of the Unified Service Query Language*. Technical Report, http://cgi.di.uoa.gr/~michaelp/TR/usql-1.0-spec.pdf
2. Tsalgatidou, A., Athanasopoulos, G., Pantazoglou, M., et al. (2006) *Generic Service Model Specification*. Technical Report, http://cgi.di.uoa.gr/~gathanas/TR/gesmo-1.0-report.pdf

3. W3C Working Group (2004) *Web Services Architecture.* Note 11 February 2004, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
4. Eiter, T. et al. (2001) *Matchmaking for Structured Objects*. In Proc. of the Third International Conference on Data Warehousing and Knowledge Discovery, Lecture Notes In Computer Science, Vol. 2114, 186-194
5. Srinivasan, N., Paolucci, M., and Sycara, K. (2004) *An Efficient Algorithm for OWL-S Based Semantic Search in UDDI*. Semantic Web Services and Web Process Composition: First International Workshop (SWSWPC), San Diego, CA, USA
6. Li Kuang, Shuiguang Deng et al. (2005) *Exploring Semantic Technologies in Service Matchmaking*, Third European Conference on Web Services (ECOWS'05), 226-234
7. Bramantoro, A. et al. (2005) *A Semantic Distance Measure for Matching Web Services*. In Proc. of the Web Information Systems Engineering – WISE 2005 Workshops: WISE 2005 International Workshops, New York, NY, USA, 217-226
8. Klein, M., Bernstein, A. (2004) *Towards High-Precision Service Retrieval*. IEEE Internet Computing, 8 (1), 30-36, Jan/Feb, 2004.
9. Xin Dong, Alon Halevy, et al. (2004) *Similarity Search for Web Services*. In Proc. of VLDB, Canada
10. Taher, L. et al. (2005) *Establishing Association between QoS Properties in Service Oriented Architecture*. In Proc. of the International Conference on Next Generation Web Services Practices (NWeSP'05), 163-168,
11. Ribeiro, C. et al. (2006) *An Ontological Approach for Personalized Services*. 20th International Conference on Advanced Information Networking and Applications, Vol. 2, 729-733,
12. Sycara, K. et al. (1999) *Dynamic service matchmaking among agents in open information environments*. ACM SIGMOD Record 28 (1), Special Issue on Semantic Interoperability in Global Information Systems, 47–53
13. Tsalgatidou, A. et al. (2006) *Developing Scientific Workflows from Heterogeneous Services*. SIGMOD Record, Vol. 35 (2), 22-28
14. SODIUM Project, http://www.atc.gr/sodium
15. Mehran Sahami, Vibhu Mittal et al. (2004) *The Happy Searcher: Challenges in Web Information Retrieval*. Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI)
16. Lin, D. (1998) *An information-theoretic definition of similarity*, in International Conference on Machine Learning