Discovering Web Services and JXTA Peer-to-Peer Services in a Unified Manner

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. Web services constitute the most prevailing instantiation of the service-oriented computing paradigm. Recently however, representatives of other computing technologies, such as peer-to-peer (p2p), have also adopted the service-oriented approach and expose functionality as services. Thus the service-oriented community could be greatly assisted, if these heterogeneous services were integrated and composed. A key towards achieving this integration is the establishment of a unified approach in service discovery. In this paper, we describe some features of a unified service query language and focus on its associated engine, which is used to discover web and p2p services in a unified manner. We exemplify how our unified approach is applied in the case of web and p2p service discovery in UDDI and JXTA, respectively. Additionally, we demonstrate how our service search engine is able to process heterogeneous service advertisements and thus to exploit the advertised syntactic, semantic, and quality-of-service properties during matchmaking.

1 Introduction

The service-oriented computing (SOC) paradigm has been successfully instantiated by the technology of web services. To date, most of the core aspects of web services have been standardized and, specifically with regard to their discovery, the *Universal Description, Discovery and Integration (UDDI)* [1] specification has been established as the preferred model of choice. Recently however, other types of services have also emerged such as peer-to-peer (p2p) services [2], fostering a new model for service sharing, discovery and reuse. Among the most well known p2p technologies currently supporting the notion of service is JXTA [3], an open peer-to-peer infrastructure which enables any connected device on the network to act as a *peer* and interact with other peers. Peers in a JXTA network are expected to interact through the services they offer/consume. Peers are organized in *peer groups,* where each peer group establishes its own policies and a set of services that all peer members should implement. Usually, peer groups are used to organize peers offering services in a specific application domain.

The established p2p infrastructure and core services of JXTA have been used in a number of cases to deploy, publish and compose p2p services. In [4], a distributed and decentralized market of p2p services was proposed, also facilitating their automatic

A. Dan and W. Lamersdorf (Eds.): ICSOC 2006, LNCS 4294, pp. 104 – 115, 2006.

composition. In [5], an approach was proposed for the semantic annotation of p2p services that could assist their automatic discovery and selection. Utilized from a different point of view, the p2p architecture was also used as the underlying infrastructure for grouping service registries into domain-specific federations [6]. Such organization provided a significant enhancement to the course of service discovery.

Even though many well known p2p technologies (e.g. [19] [20]) have not yet embraced the service-oriented architecture, the results of the aforementioned efforts could provide a strong motivation for doing so in the near future. Hence, there is an emerging need for the integration and interoperability of web and p2p services technologies. A significant step towards achieving such integration involves the establishment of a unified approach in service discovery. Currently, the existing web or p2p services can be discovered only through the underlying discovery mechanisms of the registry or the p2p network where they have been published. Thus, developers are either confined to search in a specific type of registry / network, or they are forced to employ separately the different approaches and mechanisms in order to locate services which are appropriate for their application.

In this paper, we propose a solution for discovering web and p2p services in a unified way. Our solution comprises a query language which supports the creation of queries for discovering heterogeneous services in a unified manner and its associated search engine, which tackles the heterogeneity among the existing web and p2p service discovery mechanisms and description protocols. Among the key contributions of the search engine, which is the main focus of this paper, are: (1) the provision of a unified search interface, which alleviates requesters from the burden of conducting separate service lookups in the various heterogeneous registries and p2p networks; (2) the established level of abstraction, which hides the underlying complexity and heterogeneity from the users; (3) the ability to support existing and emerging standards in service description and discovery.

Briefly, the rest of the paper is structured as follows: in Section 2, we describe a motivating scenario which underlines the need for integration of web and p2p services and also highlights the heterogeneity that hinders their unified discovery; in Section 3, we briefly describe the *Unified Service Query Language (USQL)*, which is used by our search engine for the formulation of the queries and their corresponding responses; Section 4 describes the architecture and some of the main components of the search engine; in Section 5, we demonstrate how the engine is used to discover web and p2p services in UDDI registries and JXTA networks, respectively; Section 6 compares our approach to related work and, finally, we conclude in Section 7 with a discussion on future work.

2 Motivating Scenario

In order to reveal the need for integration of web and p2p services, let us consider the following scenario from the domain of Healthcare.

The IT department of a private clinic has decided to develop a service-oriented application to enable direct interactions between doctors, patients, as well as other partners. The clinic has already established partnerships with external doctors and the

IT departments of other hospitals. Specifically, a p2p network has been established to support communication and exchange of data between the clinic and external doctors, while the partner hospitals offer a number of specialized web services to the clinic. Fig. 1 depicts an excerpt of this application, where a second opinion is requested for a specific medical episode.

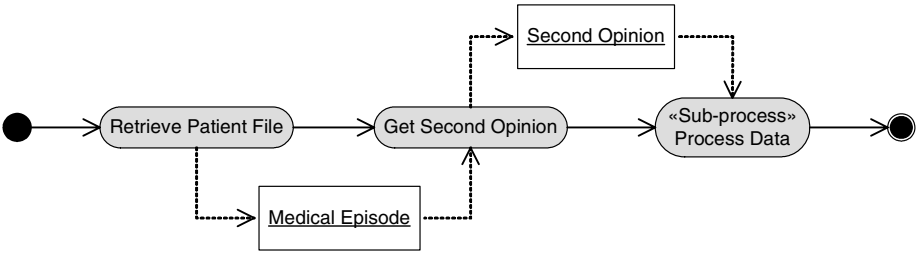

Fig. 1. A service composition requiring the integration of web and p2p services

In the above example, the patient file retrieval functionality could be offered by a web service, while doctors could communicate and exchange second opinions on specific medical incidents with the use of specialized p2p services running on their PDAs. Alternatively, partner hospitals could provide web services which offer diagnoses for specific medical episodes.

In order to implement the above service composition, the developers of the clinic's IT department have to first discover the required services from the established registries and the p2p network. Alas, the current state of the art produces a number of implications: (1) the IT department has to use separate discovery tools, which increase the development cost; (2) the developers need to acquire thorough knowledge on the technical details of the underlying discovery mechanisms and protocols, and thus fail to focus on the business part of the application.

The scenario reveals the need for integration of web and p2p services and, moreover, shows that a unified approach towards the discovery of such services would very much simplify and facilitate the work of developers. In the following sections, we describe how our search engine addresses these issues. First, we provide a very brief description of the language used by the search engine for the formulation of the queries and their respective responses.

3 The Unified Service Query Language (USQL)

The *Unified Service Query Language (USQL)* is an XML-based language enabling requesters to create meaningful queries for heterogeneous services in a unified manner, while at the same time it keeps technical details transparent. The USQL specification defines two types of messages, namely the *USQLRequest* and *USQLResponse*. To better capture real-world requirements, the language blends the flavors of syntactic, semantic and quality-of-service (QoS) search criteria. Moreover, it defines a set of operators, which can be explicitly applied to the search criteria and determine the matchmaking process. This departure is particularly useful when

applying service discovery at design time, where requirements should be expressed in a more relaxed fashion.

The snippet below illustrates a USQL request in accordance to the motivating scenario discussed in Section 2.

```
<USQL version="1.0" xmlns="urn:sodium:USQL">
 <USQLRequest>
  <ViewAdditionalProperties>
   <property>Availability</property>
  </ViewAdditionalProperties>
  <Where>
   <Service>
    <ServiceDescription valueIs="contain"> medical diagnosis</ServiceDescription>
    <ServiceDomain ontologyURI="http://onthealth#">Healthcare</ServiceDomain>
    <Operation>
     <Inputs><input>
       <type>http://www.w3.org/2001/XMLSchema#string</type>
       <semantics ontologyURI="http://onthealth#">MedicalEpisode</semantics>
      </input>
     </Inputs>
     <Outputs><output>
       <type>http://www.w3.org/2001/XMLSchema#string</type>
       <semantics ontologyURI="http://onthealth#">Diagnosis</semantics>
      </output>
     </Outputs>
     <QoS><Availability valueIs="equalOrGreater">0.9999</Availability></QoS>
    </Operation>
   </Service>
  </Where>
  <OrderBy direction="descending">Availability</OrderBy>
 </USQLRequest>
</USQL>
```

Fig. 2. A USQL request for *"get second opinion"* services

The query contains a number of syntactic, semantic and QoS requirements at various levels. Specifically, the requester is looking for *"medical diagnosis"* services in the domain of *Healthcare*. The desired operation should accept a string as input (the *medical episode*) and return a string as output (the *diagnosis*). Due to its very nature, the service should be *at least 99.99% available*. The requester has specified that the availability property should be included in the matching services (with the use of the <*ViewAdditionalProperties*> element), and moreover its value should be used for sorting the results (via the <*OrderBy*> element).

A closer look to the USQL request example reveals that all requirements were specified in a service type-agnostic manner. Indeed, the message contains no indication or requirement regarding the type of the candidate service(s). Moreover, requirements were expressed at a relatively high level, based on the intuitive knowledge of what is required for the specific task. No technical details were required

or imposed by the USQL language in formulating the request, besides the need for a basic knowledge of XML.

For the sake of brevity, we refer to [7] for a detailed description of the various structures and elements of the USQL language. Nevertheless, the provided information is considered adequate for the purposes of this paper, allowing us to proceed with the description of our service search engine.

4 The Unified Service Search Engine

The *Unified Service Search Engine* is an extensible framework used for applying service discovery in heterogeneous registries and networks. It is characterized by an open architecture enabling the smooth accommodation of various registry and service description standards, for the purposes of service discovery and matchmaking. More specifically, plug-ins are used for supporting access to the various service registries and networks, while appropriate document handlers are introduced to deal with the various syntactic, semantic and QoS service advertisements. The engine was briefly discussed in [8] and [10]; here, we will elaborate on the functionality of its various components and provide technical details regarding its implementation.

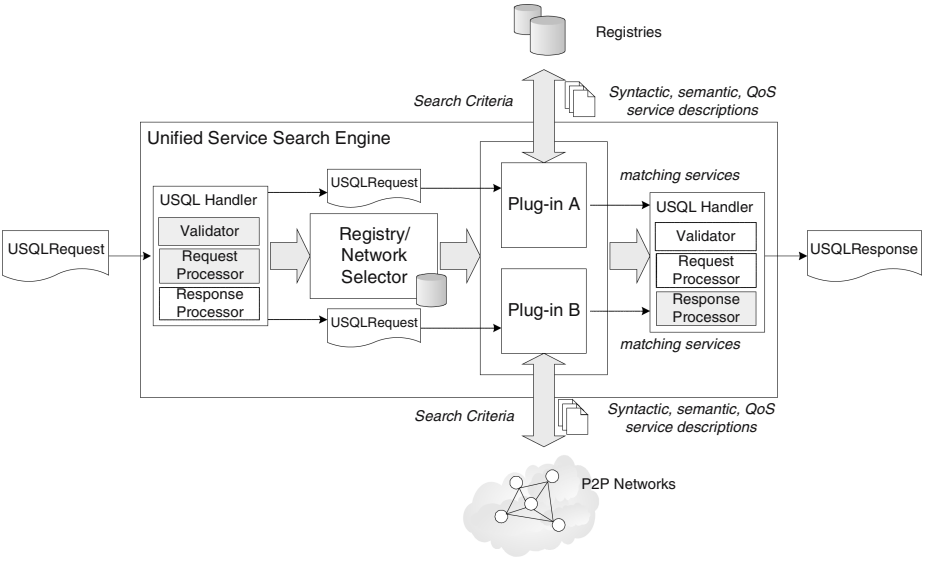

Fig. 3. Basic components of the service search engine

Fig. 3 depicts the internal structure of the search engine. Upon receiving a USQL request, the engine employs the *USQL Handler* to validate it against the USQL schema. The USQL Handler is divided into three logical parts: the *Validator*, responsible for the validation of USQL messages; the *Request Processor*, responsible for processing the content of USQL request messages; and the *Response Processor*, responsible for constructing and properly formatting the USQL response messages.

The USQL Handler component contributes significantly to the overall flexibility and maintainability of the search engine; it abstracts the rest of the components from language-specific details, thus making them resilient to potential changes in the USQL specification.

After the USQL request has been found to be valid, the request processor is activated to extract the specified service domain value from the message. The specified domain is then used by the *Registry Selector* component in identifying the target registries and/or networks for the query. As it was described in [10], the engine makes use of an upper ontology –implemented with the use of OWL (see http://www.w3.org/2004/OWL/)– which associates registries with application domains. The ontology is instantiated by a forest of domains (there is a tree for each addressed domain); also, there are registry and p2p network instances (both instantiating the *Registry* class in the upper ontology), each one of which is associated with one or more domains, and a set of related properties that are stored by the engine. These properties include the id of the plug-in to be used, along with other parameters necessary for successfully accessing the respective registry or network (e.g. JXTA peer groups might require authentication for a peer to be able to join). Note that, maintaining the ontology's instances and associating registries with domains are human-triggered tasks and form part of the search engine's configuration process.

Having identified the target registries and/or networks, the search engine configures and instantiates the respective *Plug-ins* which accept the USQL request as input and run in separate threads, thus allowing for a form of parallelism during the execution of the query. This multi-thread implementation inside the engine contributes to the improvement of its overall performance. To better explain how each registry plug-in works, we illustrate its internal structure in Fig. 4:

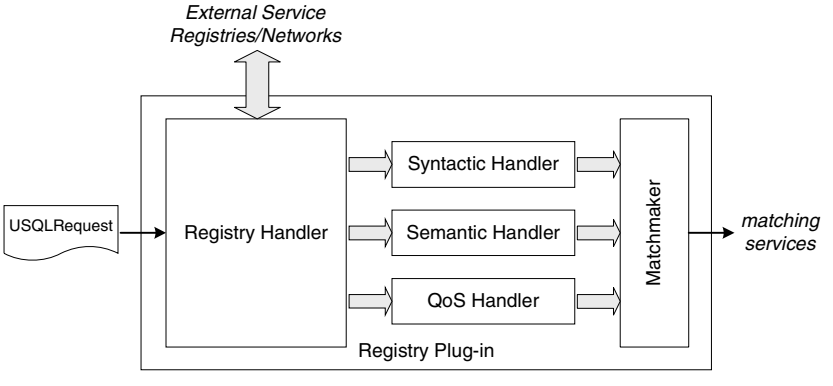

Fig. 4. Internal structure of the search engine's registry plug-ins

The *Registry Handler* component is responsible for extracting the registry-supported search criteria from the original USQL request and utilizes the specific registry type-supported discovery mechanisms and APIs to find the requested services. The process of querying the registry results in a set of service advertisements which are processed by the appropriate *Syntactic, Semantic,* and *QoS Handlers* to

extract the values of the properties that were constrained in the USQL request. Thanks to the decoupling of syntactic, semantic and QoS service description handling from the rest of the plug-in, the latter can be seamlessly extended and use different document handlers in many combinations. In this way, the search engine is capable of dealing with the various heterogeneous service description protocols.

Next, the registry plug-in employs the USQL *Matchmaker* in order to apply extended, semantically enhanced and QoS-based matchmaking to each service. The matchmaker implements a sophisticated matchmaking algorithm [9] which however goes beyond the scope of this paper. Briefly described, the algorithm calculates the overall degree of match for a given service and its operations, based on the individual degrees of match of each specified requirement. The degree of match value is a normalized float number ranging between 0 and 1. Going back to Fig. 3, the outcome of the matchmaking process, i.e. the matching services, is forwarded to the USQL Handler component, which employs the response processor to consolidate the output from all registry plug-ins into a single USQL response message.

5 Example: Unified Service Discovery in UDDI & JXTA

In accordance to the use case described in Section 2, in the following paragraphs we will demonstrate how our service search engine applies service discovery in UDDI and JXTA for *"get second opinion"* services, with the use of a single USQL request (the one that was described in Section 3). In this example, we assume that the established p2p network between the clinic and the external doctors is based on JXTA, while the web services being offered by the clinic's partners have been published to a UDDI registry. Moreover, all web and p2p services have been described with the use of WSDL-S (see http://www.w3.org/Submission/WSDL-S/) and WS-QoS [18], whilst the UDDI registry and the JXTA network have been associated with the Healthcare domain by the search engine's administrator.

5.1 Web Service Discovery in UDDI

The UDDI specifications define a set of protocols and APIs for publishing information regarding businesses and the services they offer, as well as for querying such data. The default UDDI query mechanism supports primarily keywords-based queries where only syntactic requirements can be processed. Furthermore, search criteria can be applied only at the service level and thus operation and input/output related requirements cannot be processed. The UDDI specifications partially cater for these defects, by defining an extension point, the *tModel* structure, which can be used to reference external information (e.g. WSDL or WSDL-S service descriptions). The use of the tModel facility in service discovery with UDDI is described in [11]. Our approach also exploits tModels, as we will see next.

The search engine gains access and queries the UDDI registry that has been associated with the Healthcare domain, by employing the respective UDDI plug-in. If the USQL request contains criteria which are supported by the primitive discovery mechanism of UDDI, such as the service name/description or the service provider,

these are used accordingly to narrow the lookup range. The query yields a number of tModels containing references to the WSDL-S descriptions of the published web services, as shown in the example below:

```
<tModel ...>
  <overviewDoc>
    <overviewURL>WSDL-S document URL here</overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference tModelKey="..." keyName="uddi-org:types" keyValue="wsdlSpec"/>
  </categoryBag>
</tModel>
```

Fig. 5. An example tModel structure with reference to an external WSDL-S document

These descriptions are retrieved and parsed with the use of the appropriate WSDL-S document handler, employed by the UDDI plug-in of the search engine. In a similar way, the WS-QoS document handler provided by the search engine is used to parse the referenced WS-QoS offers included in the WSDL-S documents. The extracted information is mapped to a unified, USQL-like service advertisement according to the rules given in Table 1, which is then dispatched to the USQL matchmaker component along with the USQL request for matchmaking.

Table 1. Rules for mapping WSDL-S & WS-QoS to USQL

WSDL, WSDL-S & WS-QoS	USQL
wsdl:service @name	Service /ServiceName
wsdl:operation /wsdl:input /wsdl:output @name	Service/Operation /Inputs /Outputs /name
wsdl:message/wsdl:part @name @type @wssem:modelReference	Service/Operation/Inputs/input Service/Operation/Outputs/output /name /type /semantics
wsqos:qosOffer /defaultQoSInfo/serverQoSMetrics/availability /defaultQoSInfo/serverQoSMetrics/reliability /defaultQoSInfo/serverQoSMetrics/processingTime	Service/Operation/QoS /Availability /Reliability /ProcessingTime

5.2 P2P Service Discovery in JXTA

Services in a JXTA network are advertised through a specific type of XML-based advertisement, namely the *ModuleSpecAdvertisement (MSA)*, which provides limited information regarding the service, the service provider, etc. Nevertheless, as it has already been proposed in [5], JXTA service advertisements can be extended to

support rich-content service descriptions, and thus substantially facilitate the task of service discovery. Our approach takes advantage of this extensibility in order to perform advanced service discovery in JXTA networks.

Upon its instantiation, the JXTA plug-in provided by our search engine – acting as a minimal edge peer – joins the peer group specified by configuration and submits a *"getRemoteAdvertisements"* query to the peer group's *rendezvous peer(s)*, by using the peer group's established discovery service. These special types of super peers maintain indices of peers and advertisements in the peer group, which they use in order to propagate the query to the appropriate peer(s). Like in the case of UDDI, criteria such as service name / description or provider can be used to narrow the lookup range. The rendezvous peers respond by sending to the plug-in the MSAs which were found to meet the query. Similar to the tModels, the MSAs contain links to WSDL-S documents, as the following snippet illustrates.

```
<jxta:MSA xmlns:jxta="http://jxta.org">
  <MSID>...</MSID>
  <Name>GetDiagnosisService</Name>
  <SURI>WSDL-S document URL here</SURI>
</jxta:MSA>
```

Fig. 6. An example JXTA ModuleSpecAdvertisement (MSA)

At this point, the JXTA plug-in needs not be part of the p2p network any more and therefore disconnects. By accessing the referenced WSDL-S descriptions and applying the mapping rules described in Table 1, a USQL-like advertisement is generated for each service and is consequently checked against the USQL request by the USQL matchmaker.

5.3 Shaping the Service Discovery Results

As it was described in Section 4, the response processor consolidated the results (i.e. the matching services) from the UDDI and JXTA plug-ins and generated the USQL response shown in Fig. 7. Apparently two services were found to meet the search criteria: a JXTA p2p service and a web service. The service entries in the response appear sorted in descending order according to the value of their availability. The web service availability advertised in the respective WS-QoS offer was less than what was originally requested, resulting in a smaller degree of match. Note that, both service entities contain all the necessary information for their immediate invocation. The referenced WSDL documents provide the details and bindings of the services' operations. The binding information depends on the specific service type. For instance, the WSDL document of the JXTA service includes information regarding the JXTA pipes used for communicating with the service, while the WSDL document of the web service provides the service endpoint address, encoding style, communication protocol, etc.

```
<USQL version="1.0" xmlns="urn:sodium:USQL" xmlns:srv="urn:sodium:USQL:services">
 <USQLResponse>
  <srv:Services>
   <srv:Service type="P2PService" degreeOfMatch="1.0" networkType="JXTA">
    <srv:name>GetDiagnosis</srv:name>
    <srv:descriptionDocUrl>
     http://jemini.di.uoa.gr:8080/sodium/wsdl/SecondOpinion.wsdl
    </srv:descriptionDocUrl>
    <srv:interface name="GetDiagnosisInterface">
     <srv:Operation degreeOfMatch="1.0">
      <srv:name>getDiagnosis</srv:name>
      <Availability>0.9999</Availability>
     </srv:Operation>
    </srv:interface>
   </srv:Service>
   <srv:Service type="WebService" degreeOfMatch="0.9999">
    <srv:name>GetDiagnosisWS</srv:name>
    <srv:descriptionDocUrl>
     http://jemini.di.uoa.gr:8080/sodium/wsdl/GetDiagnosis.wsdl
    </srv:descriptionDocUrl>
    <srv:interface name="GetDiagnosisIF">
     <srv:Operation degreeOfMatch="0.9999">
      <srv:name>getMedicalDiagnosis</srv:name>
      <Availability>0.9998</Availability>
     </srv:Operation>
    </srv:interface>
   </srv:Service>
  </srv:Services>
 </USQLResponse>
</USQL>
```

Fig. 7. The USQL response containing alternative *"get second opinion"* services

This concludes our example.

6 Related Work

A lot of research has revolved around service discovery over the last years and a number of service search engines and matchmakers have been proposed. In [12], a novel search engine is described which enables searching for web service operations that are similar to a given one. The underlying idea of this approach 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. Yet, the approach does not consider existing semantic service descriptions and thus, as opposed to our search engine, it does not exploit their rich content. In [11], Paolucci et al. describe how the UDDI infrastructure can be extended to support OWL-S based semantic annotations for services. The main drawback of this approach lies in that a significant update to the UDDI specifications is required. Moreover, discovery is confined to web services only. Another framework that makes use of OWL-S for automating the matchmaking process during web service discovery is the

WSML middleware, as described in [13]. However, the proposed matchmaking algorithm seems to be bound with that specific semantic description protocol and thus is not able to apply semantic matchmaking to services described with other protocols, e.g. WSDL-S. The same shortcoming also characterizes similar efforts in JXTA service discovery, such as the Oden framework [5]. As opposed to those approaches, our service search engine remains independent from the various service description protocols. Thanks to its flexible design, it can leverage existing or emerging standards, such as OWL-S and WSDL-S, and thus it can operate in a wide range of service-oriented settings.

Integration of web services with p2p networks has been extensively examined in the sense of using a p2p infrastructure to enhance the various web service activities. In METEOR-S [6], a JXTA-based p2p network is utilized to organize web service registries, in order to facilitate the tasks of service publication and discovery. Yet, to the best of our knowledge, there is no approach other than the one presented in this paper, which attempts to integrate the web service and p2p worlds in terms of unified service discovery.

7 Concluding Summary

In this paper, we briefly described the Unified Service Query Language (USQL) and some of the functional details of our service search engine supporting the unified discovery of web and p2p services. The engine is characterized by its flexible and extensible design, which renders it capable of accommodating different discovery mechanisms and service description protocols. At the same time, the technical details are kept transparent to the user, thus simplifying the task of service discovery.

Experience has revealed a number of challenges that need to be addressed by our search engine prototype. The restriction imposed by the matchmaker as regards the use of the same ontology to semantically annotate service queries and service advertisements is planned to be overcome with the utilization of a semi-automatic ontology mapping mechanism, like the one presented in [14]. Further, we are leaning towards ultimately replacing our custom upper ontology with more standardized efforts, such as the *Suggested Upper Merged Ontology (SUMO)* [15].

The matchmaker component of our search engine employs a set of distance measure functions for the calculation of the degree of match. Similarity distance measure is a very popular technique in matchmaking and has been successfully applied to similar technological areas, such as data mining and web information retrieval [16] [17]. In the future, we plan to utilize some of the already established efforts in syntactic, semantic, and QoS matchmaking, in order to enhance the precision of our search engine. Finally, to enhance the engine's performance, we are in the process of developing a caching mechanism, which will also allow us to experiment on the engine's recall.

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 (website: http://www.atc.gr/sodium).

References

1. Organization for the Advancement of Structured Information Standards (OASIS), *Universal Description, Discovery and Integration, UDDI*. http://www.uddi.org/
2. Broekstra, J., Ehrig, M. et al. (2004) *Bibster - A Semantics-Based Bibliographic Peer-to-Peer System*. WWW'04 Workshop on Semantics in Peer-to-Peer and Grid Computing
3. Traversat, B., Arora, A. et al. (2003) *Project JXTA 2.0 Super-Peer Virtual Network*, Sun Microsystems, Inc., Palo Alta, California
4. Gerke, J., Reichl, P., Stiller, B. (2005) *Strategies for Service Composition in P2P Networks*. In Proceedings of ICETE 2005, Reading, UK
5. Elenius, D., Ingmarsson, M. (2004) *Ontology-based Service Discovery in P2P Networks*. International Workshop on Peer-to-Peer Knowledge Management, P2PKM 2004
6. Verma, K., Sivashanmugam, et al (2005) *METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services*. Journal of Information Technology and Management, Vol. 6 (1), pp. 17-39
7. Tsalgatidou, A. et al. (2006) *Specification of the Unified Service Query Language (USQL)*. Technical Report, available online at http://cgi.di.uoa.gr/~michaelp/TR/usql-1.0-spec.pdf
8. Tsalgatidou, A. et al. (2005) *Semantically Enhanced Unified Service Discovery*. W3C Workshop on Frameworks for Semantics in Web Services, Innsbruck, Austria
9. Pantazoglou, M., Tsalgatidou, A., Athanasopoulos, G. (2006) *Quantified Matchmaking of Heterogeneous Services*. In Proceedings of the 7th International Conference on Web Information Systems Engineering, WISE 2006
10. Tsalgatidou, A. et al. (2005) *Semantically Enhanced Discovery of Heterogeneous Services*. 1st International IFIP/WG12.5 Working Conference on Industrial Applications of Semantic Web, IASW2005, Jyväskylä, Finland
11. Paolucci, M. et al. (2002) *Importing the Semantic Web in UDDI*. In Proceedings of Web Services, E-business and Semantic Web Workshop
12. Dong, X., Halevy, A. et al. (2004) *Similarity Search for Web Services*. In Proc. of VLDB
13. Cibran, M. A. et al (2004) *Automatic Service Discovery and Integration using Semantic Descriptions in the Web Services Management Layer*. Proc. of 3rd Nordic Conf. on Web Services, Vaxjo, Sweden
14. Li, J. (2004) *LOM: A Lexicon-based Ontology Mapping Tool*. Performance Metrics for Intelligent Systems Workshop, PerMIS 2004, Gaithersburg, MD
15. Niles, I., Pease, A. (2001) *Towards a Standard Upper Ontology*. In Proceedings of the International Conference on Formal Ontology in Information Systems
16. Sahami, M. Mittal, V. et al. (2004) *The Happy Searcher: Challenges in Web Information Retrieval*. Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2004
17. D. Lin (1998) *An Information-Theoretic Definition of Similarity*. In International Conference on Machine Learning
18. Tian, M., Gramm, A. et al. (2004) *Efficient Selection and Monitoring of QoS-Aware Web Services with the WS-QoS Framework*. IEEE/WIC/ACM International Conference on Web Intelligence, WI 2004
19. Ion Stoica, Robert Morris et al (2001) *Chord: Scalable Peer-to-peer Lookup Service for Internet Applications*. In Proceedings of ACM SIGCOMM, San Diego, CA
20. Ben Y. Zhao, Ling Huang et al. (2003) *Tapestry: A resilient global-scale overlay for service deployment*. IEEE Journal on Selected Areas in Communications