Unified Publication and Discovery of Semantic Web Services

THOMI PILIOURA and APHRODITE TSALGATIDOU
National and Kapodistrian University of Athens

The challenge of publishing and discovering Web services has recently received lots of attention. Various solutions to this problem have been proposed which, apart from their offered advantages, suffer the following disadvantages: (i) most of them are syntactic-based, leading to poor precision and recall, (ii) they are not scalable to large numbers of services, and (iii) they are incompatible, thus yielding in cumbersome service publication and discovery. This article presents the principles, the functionality, and the design of PYRAMID-S which addresses these disadvantages by providing a scalable framework for unified publication and discovery of semantically enhanced services over heterogeneous registries. PYRAMID-S uses a hybrid peer-to-peer topology to organize Web service registries based on domains. In such a topology, each Registry retains its autonomy, meaning that it can use the publication and discovery mechanisms as well as the ontology of its choice. The viability of this approach is demonstrated through the implementation and experimental analysis of a prototype.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Algorithms, Design, Measurement

Additional Key Words and Phrases: Web service publication, Web service discovery, unified, scalable, semantic Web services, evaluation, PYRAMID-S

ACM Reference Format:

Pilioura, T. and Tsalgatidou, A. 2009. Unified publication and discovery of semantic Web services. ACM Trans. Web. 3, 3, Article 11 (June 2009), 44 pages.
DOI = 10.1145/1541822.1541826http://doi.acm.org/10.1145/1541822.1541826

1. INTRODUCTION

Web services (abbreviated WS) have emerged as a dominant set of recommendations and standards. They have marked current Web engineering methodologies and are ubiquitously supported by IT vendors and academia. The rise on the

Authors' addresses: T. Pilioura, A. Tsalgatidou, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, TYPA Buildings, Panepistimioupolis Ilisia, 157 84, Athens, Greece; email: {thomi,atsalga}@di.uoa.gr
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2009 ACM 1559-1131/2009/06-ART11 $10.00
DOI 10.1145/1541822.1541826 http://doi.acm.org/10.1145/1541822.1541826

WS consumption brought forward the problem of locating the most appropriate services to use from the vast number of available ones. Different solutions to this problem have been proposed (e.g., UDDI [2003], ebXML registry/repository [ebXML RIM 2005; ebXML RS 2005], SPiDeR [Syeda-Mahmood et al. 2005], METEOR-S [Verma et al. 2005]), each with its specific model and realization. However, the effectiveness of these solutions is limited due to the following reasons:

—A large number of these solutions are syntactic-based. This means that only syntactic information is used in the service advertisement, in the service query, and in the matchmaking process. Syntactic information describes the *interface of services* and *how* and *by whom the services are deployed*, including names, types, values, structural as well as textual descriptions. Thus, it provides information about the rules, structures, and terms that someone should use in order to communicate with a service. However, the use of only syntactic information in the service advertisements, service queries, and in the matchmaking process leads to discovery results of poor quality.

—Most of these solutions are not scalable, meaning that they are not able to scale to large numbers of services, service publishers, and service requesters. This is due to the fact that they mostly follow a centralized registry approach. In such an approach, there is a registry that works as a store of WS advertisements and as the location where service publication and discovery takes place. The scalability issue of centralized approaches is usually addressed with the help of replication (e.g., UDDI). However, replicated Registries have high operational and maintenance cost. Furthermore, they are not transparent due to the fact that updates occur only periodically.

—Current solutions are incompatible, a fact that further aggravates the previous situation. Consider, for example, the case of an international company, the subsidiaries of which provide lots of Web services stored in registries of heterogeneous types (e.g., UDDI, ebXML registry/repository, SPiDeR, or METEOR-S). If someone wants to publish a service in these registries, he or she has to understand the mechanisms supported by each registry type and then separately employ these mechanisms in order to publish the service. Similarly, in case of service discovery the user has to invest considerable time visiting numerous registries of the same or different type, understanding the way to use them, entering search criteria repeatedly, and integrating potentially heterogeneous replies. We argue that the WS publication and discovery process could be greatly facilitated if the user entered publication data or search criteria only once, that is, if the publication or discovery process took place uniformly over the various heterogeneous service publication and discovery mechanisms. This becomes even more imperative given the fact that the number of private registries is increasing, especially after the decision to close the UDDI Business Registry [UBR 2006]. Companies prefer to create their own registries, as they want to have better control over the quality of published information, more sophisticated publication policies, and more efficient discovery process.

In this article, we propose a framework for Web service publication and discovery, called PYRAMID-S, which addresses the aforementioned situation and has the following main contributions.

—unified Web service publication and discovery:
  —over heterogeneous registries, thus alleviating the users from the burden of handling the diversion between different technologies;
  —based on syntactic, semantic, and Quality of Service (QoS) information, improving in this way the precision and the recall;
—preservation of the autonomy of Web service registries by allowing the accommodation of different publication and discovery mechanisms; and
—use of a scalable infrastructure which organizes registries based on domains.

PYRAMID-S adopts a layered architecture for satisfying the requirement for scalable, unified publication and discovery over heterogeneous WS registries. The requirement for publication and discovery based on syntactic, semantic, and QoS characteristics is addressed by using ontologies and two special languages described in this article. Moreover, PYRAMID-S has been designed and implemented in a modular service-oriented manner, which allows developers to reuse certain functionality, provided as Web services, in other applications that need to perform service publication and/or discovery.

We would like to note that PYRAMID-S was introduced in Pilioura et al. [2004]. Here, we further elaborate on this framework and present the implementation and evaluation of the PYRAMID-S prototype. Thus, in the following section (Section 2), we describe the PYRAMID-S architecture; then, we present the ontologies (Section 3) followed by a description of the languages used in PYRAMID-S (Section 4). In Section 5, we present the functionality of PYRAMID-S. The PYRAMID-S design is described in Section 6, followed by an analysis of the mediation support in PYRAMID-S (Section 7). Then, we proceed with the evaluation of PYRAMID-S by presenting the PYRAMID-S prototype implementation, the experiment setup, and the evaluation results (Section 8). In Section 9, we compare our approach to related work. Finally, we conclude with a discussion section.

2. PYRAMID-S ARCHITECTURE

2.1 PYRAMID-S Layered Architecture

One of the main goals of our work is to provide a scalable framework for unified service publication and discovery. One way to address scalability is by distributing services in domain-specific registries. This enables more pertinent service discovery as the selection of a domain registry works as a first filter in the discovery process. For example, if a registry is related to the “Loan Services” and “Insurance Services” domains, it will maintain Web services specific to those domains, thus, queries for such type of services can be routed to it. As regards the requirement for unified service publication and discovery, this can be satisfied by adding a layer of unification over the heterogeneous registries.

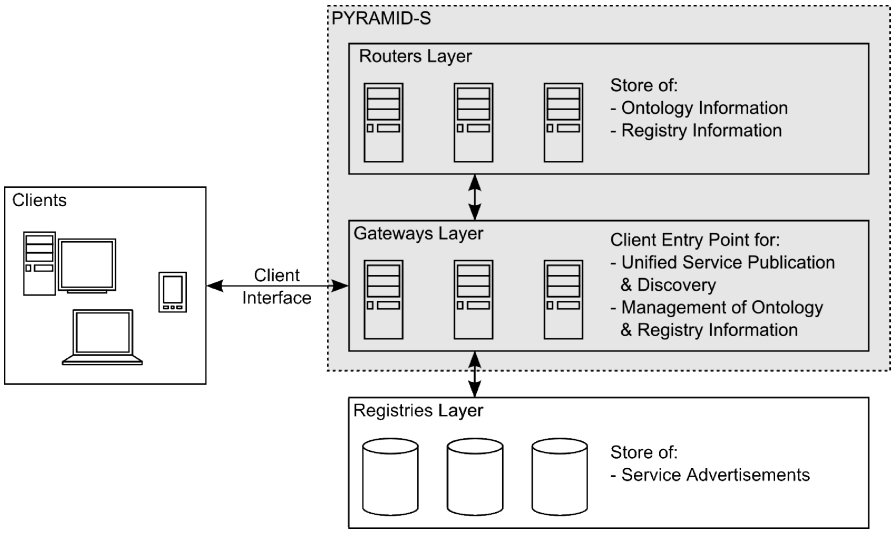

Fig. 1. The PYRAMID-S conceptual architecture.

On the basis of the preceding considerations we have proposed a framework which consists of three layers, depicted in the right part of Figure 1. The bottom layer, called the registries layer, is composed of existing registries; the other two layers (in gray background), namely the gateways and the routers layers, are introduced by PYRAMID-S in order to address the requirements for scalable and unified publication and discovery.

The *registries layer* consists of a number of registries provided by diverse registry operators. The registries are responsible for getting the service advertisements or the service queries and for performing the necessary actions. The registries may be heterogeneous due to different choices at the physical level (different DBMSs), logical level (different data models), and conceptual level (different ontologies). PYRAMID-S accommodates any kind of Web service registry, such as SPiDeR, UDDI registry, or ebXML registry, with each registry retaining its autonomy.

The two other layers of PYRAMID-S act as a metaregistry that controls and supports access to the registries, as follows.

—The *gateways layer* consists of a number of servers that are known a priori to the PYRAMID-S clients (i.e., to the service publishers, service requesters, registry operators, and domain administrators). The servers of this layer function as entry points to the PYRAMID-S system. They provide to the clients an interface for browsing and managing ontology and registry information as well as a single unified view for publication and discovery over heterogeneous registries.

—The *routers layer* consists of a number of servers that provide routing service to the gateways in order to forward the queries/advertisements entered in the gateways to the appropriate domain registries.

2.2 Peer-to-Peer Infrastructure

PYRAMID-S is based on a custom peer-to-peer (p2p) network which facilitates the communication between the distributed nodes of PYRAMID-S. We have implemented a custom p2p infrastructure instead of using an existing one (such as Gnutella [2002], Chord [Stoica et al. 2001], etc.) as we wanted to have specific features (described next) without the overhead of a full-fledged p2p system, such as time and effort to get familiarized with the more complex installation and configuration.

This infrastructure is primarily designed to facilitate peers in:

—locating one another and

—"working together" in groups and more precisely performing commonly approved actions.

To this end, some peers are identified as *replicated index peer nodes*. These nodes keep a replicated list of the participants of the p2p network; they also provide a keyword-based peer searching facility in order to facilitate peer lookup. Thus, the actions of joining and leaving the p2p network as well as peer searching take place as follows.

—*Joining the p2p Network*. A peer node, say peer A (either simple or replicated index peer node), can join the p2p network through an active replicated index peer. Thus, peer A iterates through the list of known replicated index peers[1] until it finds an active one, through which it registers to the p2p network, that is, it becomes known to all active replicated index peers. In case peer A is a replicated index peer, it also retrieves and stores the list of all currently active replicated index peers and simple peers. Therefore, every replicated index peer that enters the p2p network becomes aware of all other active nodes and shares the same state (meaning that it has the same content) with the other replicated index peers.

—*Leaving the p2p Network*. A peer node may leave the p2p network either explicitly by informing a replicated index peer or implicitly when it is shut down without having informed any other node; in the latter case, the shutdown peer is considered to have left the network only when another node discovers its unavailability and propagates this information to the whole network.

—*Peer Searching*. Simple peers can locate other peers through the replicated index peer nodes and then they can communicate with them directly.

Furthermore, all peer nodes (either simple or replicated index peer nodes) may participate in dynamic peer groups where actions are proposed by a member and performed by all members, once they have been approved, according to a decision making mechanism (briefly described in Section 6.2).

The preceding infrastructure is generic and it can be used in any application that requires the aforementioned functionality. In PYRAMID-S, the peers of

[1]Known peers are those that have been successfully contacted in the past by peer A or defined by the administrator of the initiating peer.

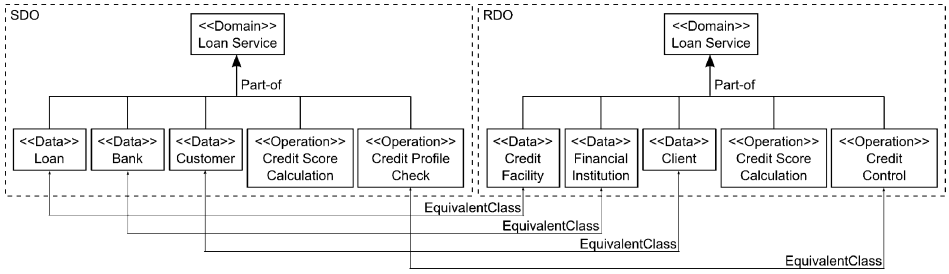

Fig. 2. A sample mapping from an SDO (left part) to an RDO (right part).

the routers layer play the role of replicated index peer nodes whereas the peers of the gateway layer are simple peers. Actually, the concept of replicated index peer nodes implies a hybrid p2p network, as elements of both pure p2p and client/server systems coexist. This p2p topology renders PYRAMID-S scalable as it allows routers and gateways to easily join and leave the p2p network. More importantly, this topology ensures that there is no single point of failure in the routers and gateways layer. Furthermore, the use of replicated index peers implies better lookup performance than pure p2p systems, as searching can be done much more efficiently in a centralized manner.

3. PYRAMID-S ONTOLOGIES

Three different types of ontologies are used by PYRAMID-S in order to address the requirement for semantics: the Standard Domain Ontology (SDO), the Registry Domain Ontology (RDO), and the Domain Classification Ontology (DCO) which are presented next.

—*Standard Domain Ontology (SDO)*. This ontology reflects abstract concepts and relationships in a particular application domain. It has two parts: the operation part and the data part. The *operation* part models major action types and thus helps to determine the type of operations that each Web service performs. For example, the operation part of an SDO for the Loans Services domain may include concepts such as CreditScoreCalculation and CreditProfileCheck. The *data* part incorporates concepts, their properties, and relations among concepts in a particular application domain. For example, the SDO for the Loans Services domain in the left part of Figure 2 may include concepts such as Loan, Customer, and InterestRate. The SDO is constructed by domain experts and it is mandatory to associate an SDO to each domain in PYRAMID-S. The SDO of a specific domain is the default ontology of the PYRAMID-S framework for that domain. The registries conforming to that SDO use this ontology for the semantic publication and querying of the Web services they hold.

—*Registry Domain Ontology (RDO)*. Registries may either adopt the SDO or use their own domain ontology (RDO). In the second case the registry operator

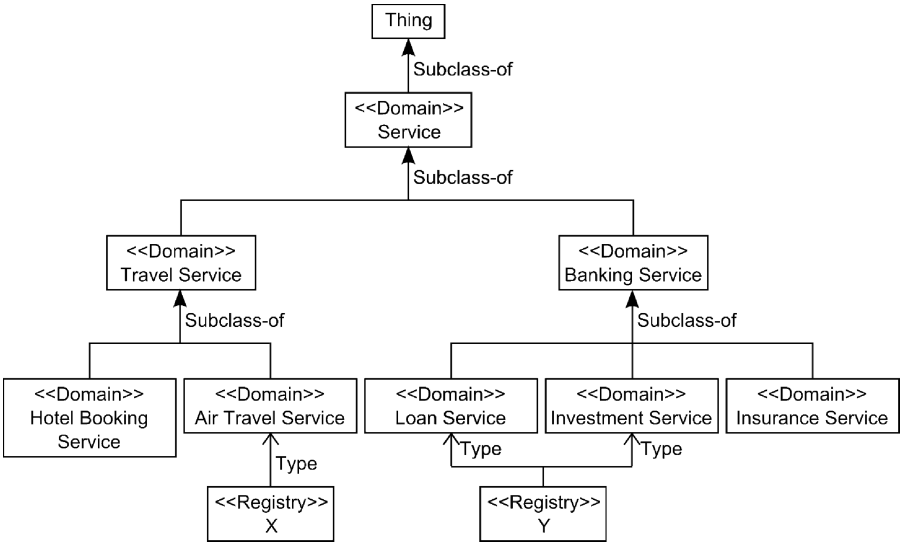

Fig. 3. A sample DCO.

has to provide a *mapping*[2] from its own ontology to the SDO (Figure 2). Mappings are not always as simple as the ones that we can see in Figure 2. For example, there are cases where a class in one ontology corresponds to a set of classes in another ontology. Ontology mapping is an area where lots of research is taking place [Bernstein and Melnik 2007; Choi et al. 2006; Doan et al. 2002; Madhavan et al. 2002] and its discussion extends beyond the scope of this article. The necessity of having RDOs and the mappings from RDOs to SDOs stems from the fact that no global enforcement on the use of ontologies is realistic in highly autonomous environments.

—*Domain Classification Ontology (DCO)*. This ontology holds information about the relationships among domains; the mappings of each registry of the PYRAMID-S to one or more domains; the relationships between domains and SDOs; and the properties of registries, such as the access URL, the registry type, the registry provider details, the access URL of the RDO to SDO mapping (in case of nonconformance to SDO), and the constraints in accessing that registry. Figure 3 shows a sample DCO in the form of a tree. Information regarding the registries in a DCO is expressed as a set of tuples T_i=<R_i, D_i, A_i>, where R_i is the access URL of a registry, D_i is a domain, and A_i are the properties of the registry. The tuples are identified by the combination of R_i and D_i, since a registry may be mapped to more than one domain. This means that for any x, y ($T_x \neq T_y \Rightarrow R_x \neq R_y \vee D_x \neq D_y$).

[2]*Ontology mapping* is the alignment of entities (concepts, attributes, relations, etc.) in one ontology with those of another ontology, so as to capture shared meaning.

4. PYRAMID-S LANGUAGES

In PYRAMID-S, the requirement for unified publication and discovery based on syntactic, semantic, and QoS characteristics is addressed by using two languages:

—the PS-WSDL, which is the PYRAMID-S extension to WSDL [2007], for describing Web services; and
—the USQL, which is the language used in PYRAMID-S for the formulation of service queries.

These languages are described in the following paragraphs.

4.1 PS-WSDL

Nowadays, the most prevalent approach for Web service description is a combination of WSDL and UDDI standards. However, the current WSDL standard describes only the syntactic aspects of a service and lacks the semantic expressivity needed to represent the service capabilities.[3] On the other hand, UDDI provides identifiers and categories to mark services using various standard taxonomies (such as related industry or geographic region).

We envisage an approach that fulfills the need for additional service information (such as QoS attributes and information about the capabilities of the service) and, moreover, it consolidates all the important information about a service (including the service provider's details) in just *one* document. Thus, we propose a language called PYRAMID-S WSDL, or in brief PS-WSDL, which is an extension to WSDL and which provides the following additional information.

—*Service Capabilities*. This information may be conveyed by semantically annotating the service operations and their respective inputs and outputs.
—*Geographic Scope of the Service*. This attribute refers to the geographic scope covered by the business functionality of the Web service rather than the location where it runs. Geographic location transparency is an important feature of Web service technology; however, there are cases in which we want to specify geographic constraints for the service. For example, a bank may want to specify that its HomeLoanService (which is a service for applying for home loans) covers only applications for houses in Greece.
—*Service Domain*. This attribute conveys information about the domain that the service belongs to such as Banking or Healthcare.
—*QoS Attributes of the Service*. This information concerns the QoS characteristics of the service, such as price, availability, reliability, and processing time.
—*Service Provider*. This information may include the name, phone, email, fax, physical address, and Web URL of the service provider.

[3]With the term *service capability* we refer to a machine-understandable description of what the service does, expressed in terms of service operations as well as inputs, outputs, preconditions, and effects of each operation.

Table I. WSDL Extensions Introduced by PS-WSDL

Requirement	WSDL 2.0 Element	Attribute/Element Introduced in PS-WSDL	Explanation
Describe WS capabilities	Input (MessageRefType) Output (MessageRefType) Operation (InterfaceOperationType)	concept (A)	Depicts (with the help of ontology concepts) the concept represented by the input/output elements or by the action that the operation performs
Describe WS domain	Interface (InterfaceType)	domain (E)	Specifies the service domain
Describe WS geographic scope	Service (ServiceType)	geoScope (E)	Specifies the geographic scope of the service
Describe WS provider	Service (ServiceType)	businessEntity(E)	Depicts information about the service provider
Describe WS QoS	Endpoint (EndpointType)	QoSMetrics (E)	Specifies QoS information about the service

Our extensions to WSDL are designed to be compliant with the existing WSDL standard. This will ensure that the extensions can be used by those who require them and can be ignored by those who do not need them. Moreover, these extensions aim at adding more expressiveness to WSDL files, with minimal modifications. Table I shows the extensions introduced to WSDL 2.0. More specifically, the first column depicts the requirement that we want to satisfy. The second one shows the WSDL 2.0 element(s) that we have redefined in order to meet the requirement. The third column records the attribute/element introduced ("A" in parentheses denotes attribute, whereas "E" in parentheses denotes element). Finally, the fourth column gives an explanation of the attribute/element introduced. We have underlined the attributes/elements that depict semantic information.

In the following we justify some of our choices regarding the WSDL extensions.

—The reason for introducing service *domain* and *geographic scope* as elements in PS-WSDL instead of attributes is that in this way we can express the fact that a service may belong to *multiple* domains and/or serve *multiple* geographic locations.

—We have introduced the *geographic scope* element in the *service* element instead of the *interface* element as the same interface may be used by a service provided by a Greek bank and by another service provided by a French bank.

—Regarding *QoSMetrics*, we have introduced them in the endpoint element and not in the binding element, as the same binding may have different QoS characteristics based on the server that serves it. For example, different Web servers may have different responses and availability depending on whether caching or load-balancing is supported or not; also different back-end systems may exhibit different performance.

```
<interface name="CreditScoreCalculationIF">
  <operation name="getCreditScore" patern= http://www.w3.org/2004/08/wsdl/in-out
     concept="http://www.nbg.gr/LoansServicesSDO.owl#Credit_Score_Calculation">
    <input messageLabel="In" element="CreditScoreCalculationRequest"
     concept="http://www.nbg.gr/LoansServicesSDO.owl#Customer_SSN" />
    <output messageLabel="Out" element="CreditScoreCalculationResponse"
     concept="http://www.nbg.gr/LoansServicesSDO.owl#Credit_Score" />
  </operation>
  <domain domainName="Commercial Banking" taxonomyURI="http://ww.naics.com/"
     domainCode="522110" />
</interface>
<service name="CreditScoreCalculation" interface="tns:CreditScoreCalculationIF">
  <endpoint name="CreditScoreCalcEndPoint"
     binding="tns:CreditScoreCalculationIFBinding"
     address="http://www.di.uoa.gr/~thomi/pyramid-s/CreditScoreCalculation">
    <QoSMetrics xmlns="http://www.wsqos.net/schemas/">
      <definition>
        <offers>
          <include url=" http://www.di.uoa.gr/~thomi/pyramid-s/QoSOffer.wsqos"/>
        </offers>
      </definition>
    </QoSMetrics>
  </endpoint>
  <geoScope locationName="Greece" geoTaxonomyURL="http://www.iso.org"
     locationCode="GR" />
  <businessEntity providerName="MyBank" providerURL="http://www.mybank.gr"
     providerAddress="Stadiou 2" providerPhoneNumber="+302107867890" />
</service>
```

Fig. 4. The PS-WSDL advertisement of the CreditScoreCalculation service.

—The *QoSMetrics* are specified by using the WS-QoS specification [Tian et al. 2004], an extensible specification developed at Freie Universitat Berlin. The WS-QoS XML schema encompasses not only the traditional transport QoS aspect (such as delay, jitter, and throughput), but also (application and Web) server (such processing time, reliability, and availability), security, transaction, SLA, and pricing-related aspects and parameters. It is easily possible to augment the schema with custom aspects and parameters. The WS-QoS XML schema allows the definition of QoS offers (that represent a minimal QoS level a service provider guarantees to supply) and QoS requirements (that express a client's QoS requirements). The definition of QoS parameters in WS-QoS may be applied both at the service and operation level. When applied at the service level, requesters implicitly express that these requirements must be fulfilled by all service operations. However, it is always implied that the usage of the QoS parameters at the operation level overrides the QoS parameters at the service level.

Figure 4 depicts part of the PS-WSDL advertisement of a CreditScoreCalculation service. A test case ontology for the domain of Loan Services is used in order to annotate the operation, input, and output elements. The NAICS taxonomy [NAICS 2007] is used to specify the service domain. The QoSMetrics element is used for specifying QoS properties of the service by using the WS-QoS specification. Elements of the Geographic Classification System (ISO 3166-1999) are used to depict the geographic scope of the service. Finally, the businessEntity element contains information about the service provider.

As it is possible to question the need for introducing a new language instead of using SAWSDL [2006], which is the initiative that mostly resembles PS-WSDL, we would like to point out the following. When we started the development of

Table II. USQL Search Criteria

Level	Criterion	Element	Description Type
Service	Name	Enables search according to the desired service name.	Syntactic
	Description	Enables search according to the desired description.	Syntactic
	Provider	Enables search according to the desired provider name.	Syntactic
	Category	Expresses the desired categorization of a service, based on a taxonomy scheme.	Syntactic
	Domain	Semantically expresses the desired service domain.	Semantic
Operation	Name	Enables search based on the desired operation's name.	Syntactic
	Description	Enables search according to the operation description.	Syntactic
	Semantics	Semantically expresses the desired functionality.	Semantic
	QoS	Extensible element of operation-level QoS criteria.	QoS
MessagePart	Name	Enables search based on the syntactic name of the desired operation's input/output parameters.	Syntactic
	Description	Enables search based on the syntactic description of the operation input/output parameters.	Syntactic
	Type	Enables search based on the syntactic type of the desired operation's input/output parameters.	Syntactic
	Semantics	Semantically expresses input/output parameters of the desired operation.	Semantic

PS-WSDL, SAWSDL was not yet released. When the latter was released, we realized that the provided extensions could not cover all the requirements of PYRAMID-S. Specifically, the extensions introduced by SAWSDL are limited to adding semantics to the *abstract* part of a service declaration. Thus, we decided to continue with the development of PS-WSDL and extend the *implementation* part of WSDL by adding information about the geographic scope, the provider, and the QoS properties of the service.

4.2 Unified Service Query Language (USQL)

USQL is the language used in PYRAMID-S for service discovery. USQL is an XML-based language which enables the formulation of queries containing syntactic, semantic, and QoS service requirements, allowing requesters to express their needs for heterogeneous (Web, p2p, or grid) services in a unified, efficient, and consistent way. Besides the syntactic, semantic, and QoS elements defined by the language, a set of operators is also provided; syntactic, semantic, and QoS operators are applied to the values of the respective elements within the course of the matchmaking process.

Table II depicts the list of search criteria defined in the USQL specification whereas Table III depicts the USQL search operators. In Figure 5, the requester

Table III. USQL Search Operators

Type	Operator	Description
Syntactic	`equal`	The value of the target element must be syntactically equal to the one specified.
	`contain`	The value of the target element must contain the one specified (equivalent to wildcards).
	`notEqual`	The value of the target element must not be syntactically equal to the one specified.
	`notContain`	The value of the target element must not contain the one specified.
Semantic	`exact`	The value of the element in the request must match exactly the value of the corresponding element in the advertisement.
	`abstraction`	The value of the element in the request must be subsumed the value of the corresponding element in the advertisement.
	`extension`	The value of the element in the request (or one of its equivalent synonyms) must subsume the value of the corresponding element in the advertisement.
	`sibling`	The value of the element in the request must have the same parent as the value of the corresponding element in the advertisement.
QoS	`equal`	The value of the target element must be equal to the one specified.
	`notEqual`	The value of the target element must not be equal to the one specified.
	`greater`	The value of the target element must be greater than the one specified.
	`less`	The value of the target element must be less than the one specified.
	`equalOrGreater`	The value of the target element must be either equal to or greater than the specified one.
	`equalOrLess`	The value of the target element must be either equal to or less than the one specified.

```
<USQL version="1.0" xmlns="urn:sodium:USQL">
  <USQLRequest>
    <Where>
      <Service serviceType="WebService">
        <Operation>
          <Inputs>
            <input>
              <semantics ontologyURI=" http://www.nbg.gr/LoansServicesSDO.owl">
                http://www.nbg.gr/LoansServicesSDO.owl#Customer_SSN
              </semantics>
            </input>
          </Inputs>
          <Outputs>
            <output>
              <semantics ontologyURI=" http://www.nbg.gr/LoansServicesSDO.owl">
                http://www.nbg.gr/LoansServicesSDO.owl#Credit_Score
              </semantics>
            </output>
          </Outputs>
        </Operation>
      </Service>
    </Where>
  </USQLRequest>
</USQL>
```

Fig. 5. An example USQL request.

asks for Web services offering an operation taking as input the social security number of the customer, and returning as output his credit score. We refer the reader to Tsalgatidou et al. [2006] for a detailed description of USQL version 1.0 which is used in PYRAMID-S.

The rational of our decision to use USQL instead of using a standard query language, such as XQuery [Boag et al. 2006] or SPARQL [Prud'hommeaux and Seaborne 2006], is that the latter are more generic languages whereas USQL has been tailored to the needs of service discovery. Specifically, XQuery is an expression language designed for processing and formatting XML data which could be used in a complementary fashion by the implementations of the USQL specification, in order to process XML-based service advertisements according to the requirements expressed in USQL queries. Similarly, SPARQL is a language used for constructing queries against resources semantically described with the use of RDF; as such it could also be used in a complementary manner by USQL implementations, in order to retrieve information from RDF-based service advertisements.

5. PYRAMID-S FUNCTIONALITY

After having presented the PYRAMID-S layered architecture, ontologies, and languages, we now proceed with the description of PYRAMID-S functionality.

The users of PYRAMID-S are either humans or agents (software programs). User actions are directed to any peer of the gateway layer, which then, depending on the user action, interacts with either a router peer or both a router peer and one or more registry peers and returns a reply. User actions may vary from service queries or advertisements to modifications of the Domain Classification Ontology (DCO), depending on the role of the user. PYRAMID-S distinguishes three different types of users.

—simple users, who publish and discover Web services;
—registry operators, who can add/remove registries from PYRAMID-S and update registry information; and
—domain administrators, who can update the DCO by adding new domains.

Figure 6 depicts a UML use case diagram that shows the main functionality of PYRAMID-S and the three types of users who interact with it. Browsing and searching through the DCO and Standard Domain Ontologies (SDOs), represented by the ontology navigation use case, is available to all types of users and greatly facilitates the rest of their actions, as will be apparent later on. These actions are represented by the rest of the use cases depicted in Figure 6 and are described in the following paragraphs.

Service publication. As we can see in the UML activity diagram depicted in Figure 7, when a user wishes to register a service in PYRAMID-S, he or she contacts a gateway peer and first selects the domain for publication; the latter is achieved either by searching by domain name or by navigating the DCO.

Afterwards, the service publisher loads the WSDL file describing the service, annotates it with the concepts of the SDO corresponding to the selected

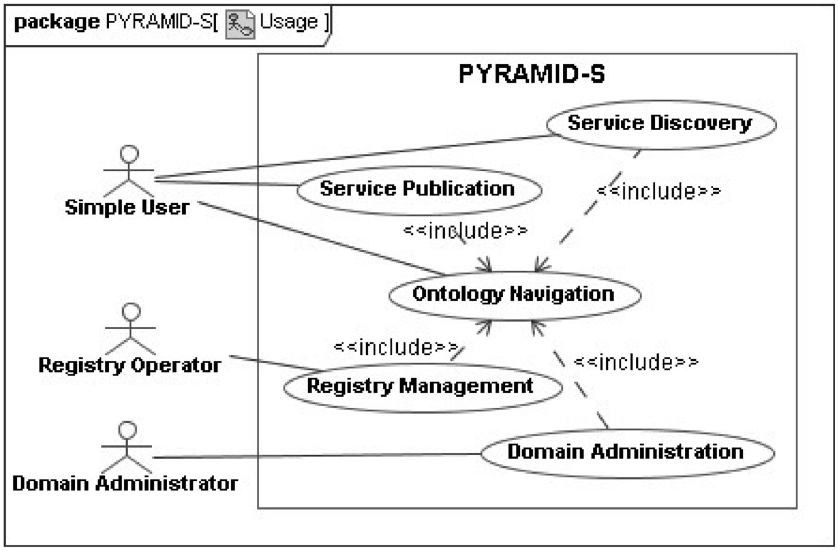

Fig. 6. Use case diagram for PYRAMID-S.

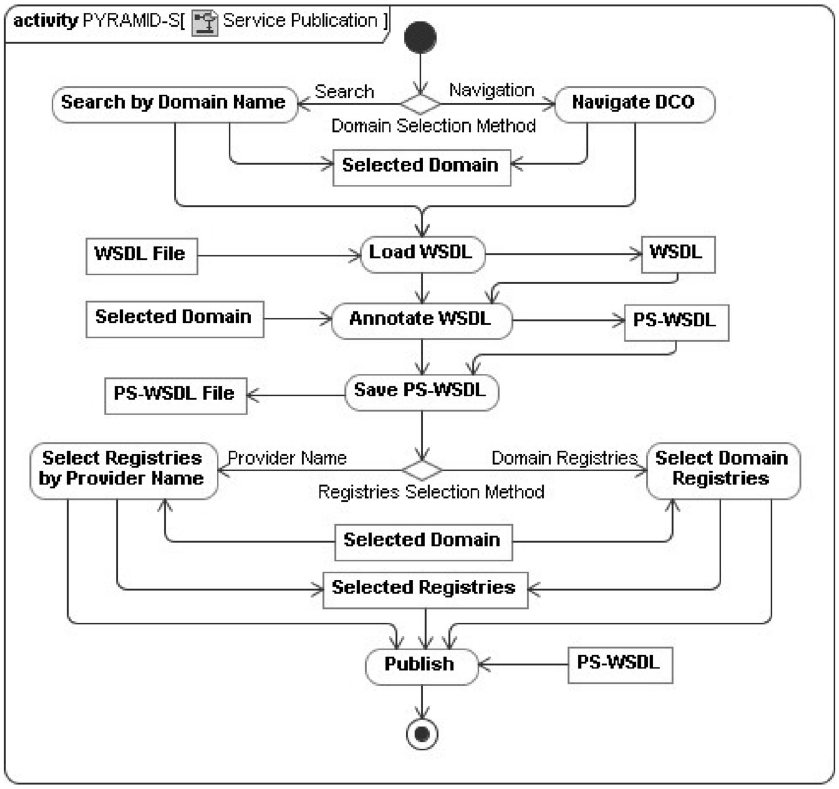

Fig. 7. Activity diagram for service publication.

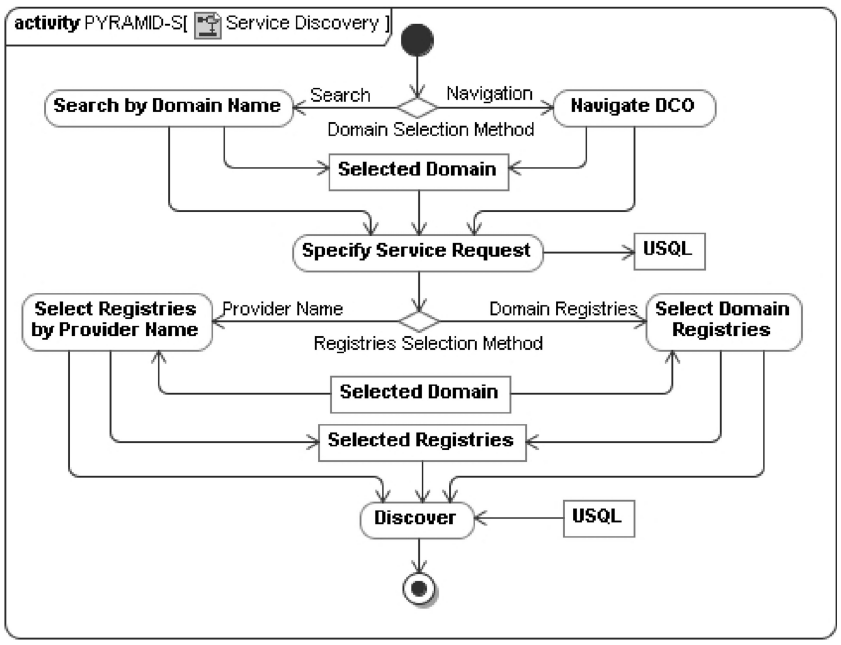

Fig. 8. Activity diagram for service discovery.

domain, and then saves the resulting PS-WSDL file. Then, the publisher may designate the registries for publication either by selecting all or several of the registries of the previously selected domain or by selecting the registries of a specific provider in the domain. Finally, the publisher proceeds with the publication of the PS-WSDL to the selected registries. Then, the Gateway contacts the appropriate PYRAMID-S components (described in Section 6) in order to complete the service publication. Finally, the user is informed about the result of his/her request.

Service discovery. When a user wishes to search for a service in PYRAMID-S, he or she contacts a gateway peer and first selects the domain for discovery. Afterwards, the service requester specifies his/her requirements by using concepts of the SDO corresponding to the selected domain. This results in a USQL query. Then, the requester may designate the registries for discovery. Finally, the requester proceeds with the service discovery in the selected registries based on the USQL. Then, the gateway contacts the appropriate PYRAMID-S components (described in Section 6) in order to complete the service discovery. The results of the registries are then returned to the gateway and presented to the user. Figure 8 depicts the UML activity diagram showing the user actions for service discovery in PYRAMID-S.

Registry management. Registry operators may use the respective interface provided by a gateway in order to insert/delete a registry or update its associated properties in the PYRAMID-S system. The part of the DCO that depicts the

relationships among domains is presented to the registry operator (in a tree structure) in order to associate his/her registry to the appropriate domain. The user input is translated into one of the following operations on the DCO.

—*Insert* (T_x). Based on the user input, registry R_x is related to domain D_x and provides its services with A_x properties ($T_x = <R_x, D_x, A_x>$). This operation is valid only if there is no $T_y \in$ DCO: $R_y = R_x \wedge D_y = D_x$. After the completion of the operation, DCO is DCO + $\{T_x\}$.

—*Delete* (T_x). Registry R_x is no longer related to domain D_x. This operation is valid only if $T_x \in$ DCO. After the completion of the operation DCO is DCO–$\{T_x\}$.

—*Update* (T_x,A_x). The properties of registry R_x for domain D_x are updated to A_x. This operation is valid only if $T_x \in$ DCO. After the completion of the operation DCO is DCO–$\{T_x\}$ + $\{(R_x, D_x, A_x)\}$.

Domain administration. Domain administrators may update the DCO with the addition of new domains. Domain renaming or deletion in the DCO are not allowed, as they would introduce inconsistency regarding registered registries and services.

6. PYRAMID-S DESIGN

In this section, we present the design of gateways and routers and describe how the aforementioned functionality is offered. At this point, we would like to stress that PYRAMID-S follows a service-oriented design, meaning that its functionality is provided through the definition of a number of Web services. This entails several benefits, such as separation of concerns and ability to substitute existing Web services with new ones, providing improved functionality and reusability of the functionality provided by the PYRAMID-S Web services from within other applications. Figure 9 depicts the various components of PYRAMID-S and their interactions. In the following paragraphs, we elaborate on this figure.

6.1 Gateway Peers

There are two ways of accessing PYRAMID-S: through a Web GUI or through a Web Service interface (WS interface). Each gateway peer provides both interfaces (Figure 9). By using the Web GUI, human users may perform service queries/publications or administration tasks. The Web GUI interacts with the WS interface of a gateway peer, through which the PYRAMID-S functionality becomes available. The Web GUI is enhanced with additional facilities, such as visual navigation of the DCO and SDOs. Furthermore, the gateway WS interface is publicly available and may be accessed by other client applications.

Gateway peers are simple peers which use routers' services through the *gateway* Web service. The latter utilizes four utility Web services, namely:

—the *ontology accessor*, which is responsible for performing ontology-related tasks;

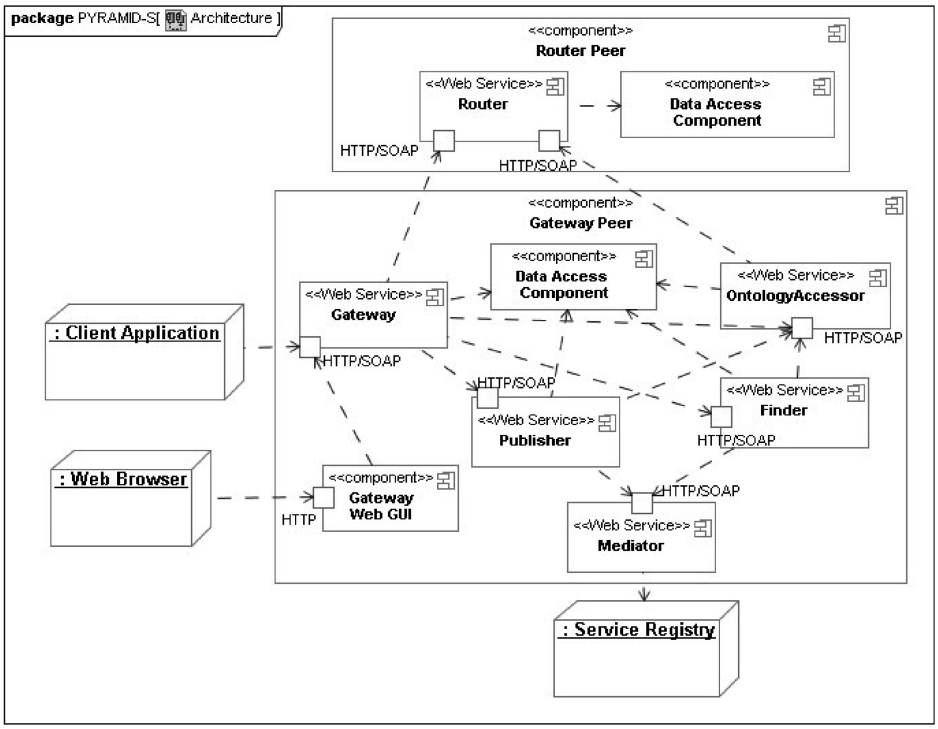

Fig. 9. Detailed design view of PYRAMID-S.

—the *mediator*, which is responsible for transforming service queries/advertisements from PYRAMID-S representations to registry-specific representations;
—the *publisher*, which is responsible for performing the service publication; and
—the *finder*, which is responsible for forwarding the service queries to the registries and for collecting the results.

Figure 9 depicts the interdependencies among these services, the functionality of which is described in detail next.

In order to perform its task, the *gateway* Web service should know the access points of the ontology accessor, the publisher, and the finder. This information is maintained in a data store.

The *ontology accessor* communicates through the p2p network with router peers in order to perform ontology-related tasks, such as retrieve the SDO, retrieve the reference to the SDO to RDO mapping, retrieve or update the DCO, retrieve the registries of a specific domain, or retrieve the type of a registry.

The *mediator* is a utility service that transforms service queries/advertisements from PYRAMID-S representations (namely PS-WSDL and USQL) to registry-specific representations. Service discovery results, deriving from registries, are also transformed reversely. Implementation of the transformation

operations depends on syntactic and semantic conventions used by the registry, as well as the support for QoS characteristics. Therefore, a distinct mediator service is needed for every type of registry participating in the PYRAMID-S framework, which may result in having more than one mediator in each gateway. Furthermore, in case a registry uses its own domain ontology instead of the standard ontology for the domain (SDO in PYRAMID-S terms), the mediator is responsible for transforming the concepts used for annotating the service advertisement/query from the one ontology to the other.

One important feature of our custom p2p infrastructure is the sharing of mediators among gateways. In case a gateway does not have one of the mediators needed for service publication or discovery in a certain registry, it can query the replicated index peer nodes in order to find gateways with the desired mediator.

The main benefits accruing from the use of mediators are the following.

—Service publishers/requesters are not required to hold technical and protocol-specific knowledge for registries.

—PYRAMID-S is rendered extensible, as it can accommodate various types of registries through the accommodation of the appropriate mediators.

Mediators are further analyzed in Section 7 due to their importance for PYRAMID-S.

The *publisher* performs a service publication given the Service Advertisement (SA) and optionally a set of selected registries for the publication. If no registries are specified, the ones that belong to the domain specified in the SA are located by the publisher through the domain classification Ontology (DCO), and they are subsequently used. The appropriate mediators are then used for translating the SA to registry interpretable forms. Thus, the publisher should know how to access the ontology accessor and the mediators. This information is maintained in a data store.

In a similar manner, the *finder* accepts Service Queries (SQ) expressed in USQL. USQL queries may either specify a set of selected registries for the discovery or explicitly specify the domain of the service that is being looked for. In the latter case, the finder uses the DCO (retrieved by the ontology accessor) to locate the registries that belong to the domain(s) specified in the SQ. Discovery requests are sent to the selected registries, after the necessary transformations are performed by the appropriate mediators. Discovery results are reversely transformed by the mediators and forwarded to the finder, which returns the aggregated result to the gateway Web service. Thus, the finder should have knowledge of the ontology accessor and the mediators. This information is maintained in a data store.

Finally, the *data access component* depicted in Figure 9 is used for accessing the data store that contains the information mentioned before.

6.2 Router Peers

Router peers play the role of replicated index peer nodes according to the principles discussed in Section 2.2. Furthermore, each router peer holds a copy of the

DCO in a local datastore, which is accessed through a *data access component*. Through the use of the DCO, the routers provide routing service to the gateways in order to forward the requests to the appropriate domain registries. Each router peer comprises the *router* Web service that provides the functionality of the replicated index peer node and allows retrieving and updating the DCO. Insertions and updates of the DCO (actions hereinafter) may be performed by any of the routers and are propagated to all other routers. Therefore, consistency of the DCO should be assured during conflicting actions, performed by the same or distinct routers within a short timeframe (i.e., the time needed for an operation to be propagated to the whole peer group). As may already have become apparent from the description of domain administration and registry management use cases (see Section 5), the following conflicting actions may occur.

—two or more routers are trying to insert the same domain in different DCO paths or with different attributes (such as different SDO); and
—insertions or updates are performed on the same couple of R_x and D_x, that is, on the same T_x during registry management.

In order to cater for these situations and to preserve the consistency of the DCO, routers implement a conflict resolution mechanism that uses a request/reply/notify scheme. This may be seen as a distributed lock management scheme, where each requesting router requests permission from the routers peer group (by simultaneously sending messages to all known routers), for performing an action on a specific part of the DCO. In case of a conflict, at most one router will succeed, since overall acceptance is decided upon the balance of positive and negative replies each requesting router has received. In case the conflicting routers receive equal positive replies, none of them gets the permission. The detailed description of this mechanism, which was originally presented in Pilioura et al. [2004], falls outside the scope of this article.

6.3 PYRAMID-S Component Interactions

In the following, we provide two sequence diagrams that present the interactions among the users of the PYRAMID-S system, the Web server GUI and the aforementioned Web services. Figure 10 depicts the exchange of messages among the various roles for performing a registry addition, whereas Figure 11 shows the necessary interactions for service publication. In order to minimize service publication time, we have exploited distributed, concurrent execution of inherently parallel tasks at two levels: publisher-to-mediators (messages 10 and 12 in Figure 11) and mediator-to-registries (messages 11 and 13 in Figure 11). Asynchronous, nonblocking Web service operation invocations (implemented with multithreading) enable distinct mediators and registries to work in parallel. Each span/fork level (publisher or mediator) aggregates all asynchronously returned results before proceeding. Querying process facilitates concurrent execution in a similar way.

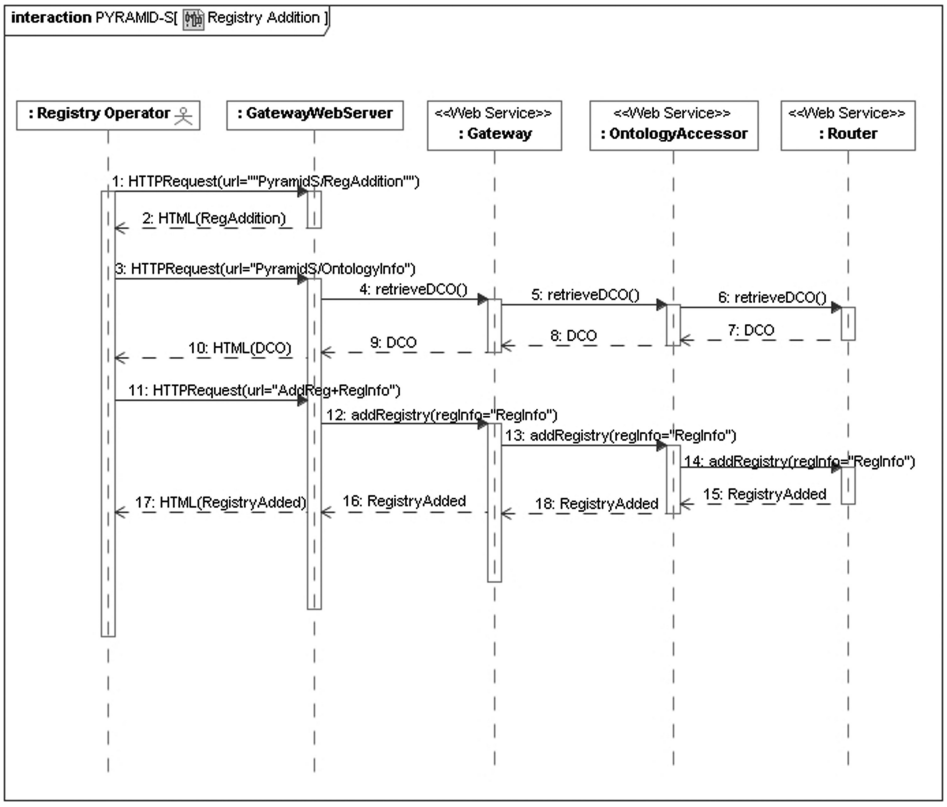

Fig. 10. Sequence diagram for registry addition.

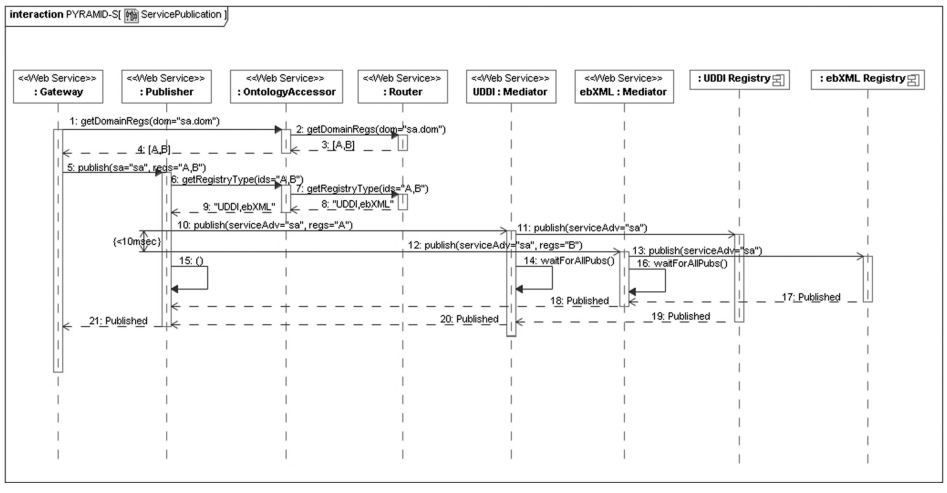

Fig. 11. Sequence diagram for service publication.

```
1: Mediator algorithm for publishing web services
2: input: PS-WSDL advertisement
3: input: list of registries of Type X
4: returns: list of booleans indicating the success of service
5: publication in each registry
6: begin
7:  set result as list of booleans
8:  set result.length as (list of registries of Type X).length
9:  set all elements of result as false
10: parse PS-WSDL advertisement
11: for every registry in the list of registries
12:   create structures for syntactic information
13:   if the registry uses own ontology then
14:     map semantic information to registry's ontology
15:  end if
16:  create structures for semantic information
17:  create structures for QoS information
18:  set result for this registry as true
19: end for
20: return result
21:end
```

Fig. 12. Mediator algorithm for publishing Web services.

7. MEDIATION SUPPORT IN PYRAMID-S

Mediators have been largely used [Garcia-Molina et al. 1997; 1995; Fensel and Bussler 2002] as an approach for integrating heterogeneous information sources. In this Section, we elaborate on the use of mediators in PYRAMID-S. We first present the publication and discovery algorithms that a mediator must implement. We then present the mappings that we have defined, upon which the mediator for UDDI and the mediator for ebXML are based in order to implement these algorithms. It is worth noting here that it is necessary to implement mediators for each type of registry that is desired to include in PYRAMID-S. Concluding this section, we discuss some issues related to mediators.

7.1 Mediator Algorithms

In PYRAMID-S, a mediator is responsible for transforming the syntactic, semantic, and QoS information of the PS-WSDL advertisement or USQL query into appropriate registry structures. The resulting structures are used for publication/discovery in the user-specified list of registries or in the list of registries found in the DCO. The communication with the registries is realized through the publication and inquiry APIs provided by the registries.

The algorithm used by a mediator for publishing the PS-WSDL advertisement in a list of registries is shown in Figure 12.

Similarly, USQL queries may be formulated by the requesters and transformed by the mediator to the appropriate registry inquiry API function calls. The algorithm used by the mediator for the WS discovery is depicted in Figure 13. As shown in this algorithm, the result consists of three sets of services. First comes the set of services satisfying both semantic and syntactic requirements specified by the requester. Second comes the sets of services satisfying only semantic requirements and then the services that meet only syntactic requirements. Whenever QoS requirements are specified in a query, their satisfaction is obligatory for a service to be returned.

```
1: Mediator algorithm for discovering web services
2: input: USQL service query
3: returns: ServiceResult /*composite structure with information about
4: services satisfying QoS requirements and
5: (a) both semantic and syntactic, (b) only
6: semantic, and (c) only syntactic requirements*/
7: begin
8:  while parsing USQL query
9:   set regs as list of registries to be queried
10:  set sem as registry structure with semantic requirements
11:  set synt as registry structure with syntactic requirements
12:  set QoS as registry structure with QoS requirements
13: end while
14: set sems as services published in regs satisfying sem and qos
15: set synts as services published in regs satisfying synt and qos
16: set result as new ServiceResult(intersection(sems,synts),sems,synts)
17: return result
18:end
```

Fig. 13. Mediator algorithm for discovering Web services.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceResult xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://tempuri.org/">
  <SemanticSyntactic/>
  <Semantic>
    <ServiceResultRow>
      <name>BusinessLoansService</name>
      <description>This document describes a web service
      for managing Business Loans</description>
      <providerName>National Bank of Greece</providerName>
      <descriptionURL>http://www.nbg.gr/wsdl/BusinessLoansService.pswsdl
      </descriptionURL>
      <accessPoints>
       <accessPoint>http://www.nbg.gr/BusinessLoansService.asmx</accessPoint>
      </accessPoints>
      <qos>
        <availability>0.77</availability>
        <reliability>0.67</reliability>
        <price>0.65</price>
      </qos>
    </ServiceResultRow>
    <ServiceResultRow>
      <name>ConsumerLoansService</name>
      <description>This document describes a web service
      for managing Consumer Loans</description>
      <providerName>Alpha Bank</providerName>
      <descriptionURL>http://www.alpha.gr/wsdl/ConsumerLoansService.pswsdl
      </descriptionURL>
      <accessPoints>
       <accessPoint>http://www.alpha.gr/ConsumerLoansService.asmx</accessPoint>
      </accessPoints>
      <qos>
        <availability>0.87</availability>
        <reliability>0.65</reliability>
        <price>0.85</price>
      </qos>
    </ServiceResultRow>
  </Semantic>
  <Syntactic/>
</ServiceResult>
```

Fig. 14. An example ServiceResult.

Figure 14 depicts an example ServiceResult for the USQL query of Figure 5. Note that only the *semantic* set of services contains elements whereas the other two sets are empty since the USQL query of Figure 5 comprises only semantic criteria. As we can see in this figure, the returned information about each service consists of the service name, the service text description, the provider

name, the URL of the PS-WSDL or the WSDL file, the access point, and the values of the QoS metrics.

7.2 The PS-WSDL/USQL to UDDI Mapping

A recommended approach for mapping WSDL descriptions to UDDI data structures is described in Colgrave and Januszewski [2004]. The goal of this technical note is to enable the automatic registration of WSDL definitions in UDDI and to enable precise and flexible UDDI queries based on specific WSDL artifacts and metadata without further recourse to the source WSDL documents. In the context of this mapping, a set of canonical tModels are defined in order to represent WSDL metadata and relationships, such as the WSDL Entity Type tModel and the XML Local Name tModel. In our work, we extend on this mapping in order to enhance UDDI version 2[4] with semantic and QoS information about services. More specifically, we have the following.

—We define some new canonical tModels that are used in conjunction with the ones specified in Colgrave and Januszewski [2004] in order to represent the PS-WSDL/USQL to UDDI mapping. These new canonical tModels are described in Section 7.2.1.

—We extend some of the mappings described in Colgrave and Januszewski [2004] and, furthermore, we define some new mappings in order to capture the additional information contained in PS-WSDL in comparison to WSDL. Section 7.2.2 presents the introduced mappings.

7.2.1 *New Canonical tModels.* Canonical tModels are used to facilitate registration of common information in UDDI. In our case, they are used to represent PS-WSDL semantic and QoS metadata as well as relationships. These tModels must be registered *once* in the UDDI registry and reused whenever these new classes of information need to be captured in UDDI service advertisements or queries. In the following, we present the tModels that we have created.

—*Interface Reference*. The WSDL Interface Reference tModel provides a mechanism to indicate that a UDDI entity has a relationship with a certain interface tModel. This can be applied, for example, to indicate that an operation tModel (see, for example, the one in Figure 15) describes an operation of a specific interface tModel.

—*Operation Reference*. This tModel provides a mechanism to indicate that a UDDI entity has a relationship with a certain operation tModel. It may be used, for example, to indicate that an interface tModel represents a WSDL interface that supports an operation described by a specific operation tModel.

—*Operation Concept*. It is used to express a relation between an operation and a concept from a specific ontology. It is to be used with operation tModels.

[4]The decision to work with UDDI version 2 was imposed by the lack of free implementations of the UDDI version 3 specification.

—*Input Concept*. It is used to express a relation between an operation's input element and an ontology concept. It is to be used with operation tModels.
—*Output Concept*. This tModel is used to express a relation between an operation's output element and an ontology concept. It is to be used with operation tModels.
—*Price*. It is used for attributing price information to a QoS tModel.
—*Availability*. It is used for attributing availability information to a QoS tModel.
—*Reliability*. It is used for attributing reliability information to a QoS tModel.
—*QoS Reference*. This tModel is used as a tag in order to denote that a binding template has an associated QoS tModel (explained in Section 7.2.2) that contains information about the QoS properties of the endpoint represented by the binding template.

7.2.2 *The Mapping Specification from PS-WSDL/USQL to UDDI.* This section summarizes the mapping specification from PS-WSDL advertisements or USQL queries to the UDDI version 2 data model.

Similarly to Colgrave and Januszewski [2004], we map each PS-WSDL artifact to a separate UDDI entity, accurately representing the "building block" design of PS-WSDL advertisements. The *interface, operation*, and *QoSMetrics* elements of PS-WSDL map to the homonym *tModel* entities. The *businessEntity* element maps to a *businessEntity* structure, the *service* element maps to a *businessService* entity, and the *endpoint* element maps to a *bindingTemplate* entity. Table IV depicts the proposed mapping. The additions we propose in relation to the UDDI technical note [Colgrave and Januszewski 2004] are marked with gray background.

In the following, we will focus on the way QoS and semantic information (domain, location, operation concepts, input concepts, and output concepts of a Web service) are captured in UDDI by using as an example the PS-WSDL advertisement of Figure 4. The domain and geographic scope information are added as keyedReferences in the categoryBag of the interface tModel and the businessService entity, respectively. The semantics of an operation and of its respective inputs and outputs are captured in the categoryBag of the associated operation tModel. Figure 15 shows the XML representation of the operation tModel for the getCreditScore operation of the service depicted in Figure 4. Apart from the semantic information, the categoryBag specifies the PS-WSDL namespace, indicates that the tModel is of type "operation", and makes reference to the interface defining this operation by using the *interface reference* canonical tModel introduced in Section 7.2.1.

The QoS information of a WS endpoint is represented in a QoS tModel, which is referenced in the binding template that represents the specific endpoint. In this tModel each QoS metric (price, availability, reliability) is represented by a *keyedReference*. The type of the QoS metric is specified by the *tModelKey* of the respective canonical tModel (described in Section 7.2.1) and its value is specified by the *keyValue*.

Table IV. Mapping PS-WSDL to UDDI

PS-WSDL	UDDI
interface	tModel (categorised as an interface)
interface namespace	keyedReference in categoryBag
interface local name	tModel name
location of WSDL document	overviewURL
service domain	keyedReference(s) in categoryBag
interface defined operations	keyedReference(s) in categoryBag
service	**businessService** (categorised as a service)
service namespace	keyedReference in categoryBag
service local name	businessService name
documentation	businessService Description
geographic scope	keyedReference(s) in categoryBag
interface implemented by this service	keyedReference in categoryBag
endpoint	**bindingTemplate**
address	accessPoint
interface implemented by endpoint	tModelInstanceInfo with tModelKey of the tModel corresponding to the interface
QoSMetrics	tModelInstanceInfo with the tModelKey of the QoSReference tModel (to denote that this binding has an associated QoS tModel) and the tModelKey of the tModel describing the QoS properties of the endpoint
businessEntity	**businessEntity**
businessEntity local name	businessEntity name
address and phone number	contact
URL	discoveryURL
operation	**tModel** (categorised as an operation)
operation namespace	keyedReference in categoryBag
operation local name	tModel name
location of WSDL document	overviewURL
operation concept	keyedReference in categoryBag
input concept(s)	keyedReference(s) in categoryBag
output concept(s)	keyedReference(s) in categoryBag
interface linked to the operation	keyedReference in categoryBag
QoSMetrics	**tModel** (categorised as QoSMetrics)
service local name + "QoSInfo"	tModel name
price	keyedReference in categoryBag
availability	keyedReference in categoryBag
reliability	keyedReference in categoryBag

Once the PS-WSDL to UDDI mapping has taken place, relevant queries may be formulated. As mentioned earlier, in PYRAMID-S we use USQL for formulating service queries that can be directed to any registry type through the appropriate mediator. The various search criteria in USQL are transformed to the appropriate UDDI inquiry functions (e.g., find_service, find_business, find_tModel) depending on the entity where the information specified by the criteria has been stored. The USQL to UDDI mapping is depicted in Table V.

Figure 16 shows the XML representation of the UDDI inquiry function that corresponds to the USQL request of Figure 5. The two tModelKeys used in the categoryBag correspond to the Input Concept and the Output Concept

```
<tModel tModelKey="uuid:D17AC770-7BA0-11DB-8770-D672BAE71F04">
  <name>getCreditScore</name>
  <overviewDoc/>
  <categoryBag>
    <keyedReference keyName="WSDL Type" keyValue="operation"
      tModelKey="uuid:6E090AFA-33E5-36EB-81B7-1CA18373F457"/>
    <keyedReference keyName="operation namespace"
      keyValue="http://www.di.uoa.gr/~thomi/ pyramid-s/CreditScoreCalculation"
      tModelKey="uuid:D01987D1-AB2E-3013-9BE2-2A66EB99D824"/>
    <keyedReference keyName="interface reference"
      keyValue="uuid:CCE36500-7BA0-11DB- A500-835E46D83C63"
      tModelKey="uuid:B8D83260-7007-11DB-B81F-C922527398E7"/>
    <keyedReference keyName="operation concept"
      keyValue="http://www.mybank.gr/LoansServicesSDO.owl#Credit_Score_Calculation"
      tModelKey="uuid:98769880-7008-11DB-B81F-ABDA6268B638"/>
    <keyedReference keyName="input concept"
      keyValue="http://www.mybank.gr/LoansServicesSDO.owl#Customer_SSN"
      tModelKey="uuid:4A8DDFB0-7009-11DB-B81F-B0A4C6BCF510"/>
    <keyedReference keyName="output concept"
      keyValue="http://www.mybank.gr/LoansServicesSDO.owl#Credit_Score"
      tModelKey="uuid:795D0F00-7009-11DB-B81F-9C74A8A2C891"/>
  </categoryBag>
</tModel>
```

Fig. 15. Operation tModel of the getCreditScore operation.

Table V. Mapping USQL to UDDI

USQL	UDDI
Service Name	find_service/name
Service Provider Name	find_business/name
Service Taxonomy	find_tModel/categoryBag/KeyedReference, find_service/categoryBag/KeyedReference
Operation Name	find_tModel/name, find_tModel/categoryBag/KeyedReference, find_service/categoryBag/KeyedReference
Operation Capability	find_tModel/categoryBag/KeyedReference, find_service/categoryBag/KeyedReference
Operation input semantics	find_tModel/categoryBag/KeyedReference, find_service/categoryBag/KeyedReference
Operation output semantics	find_tModel/categoryBag/KeyedReference, find_service/categoryBag/KeyedReference
Service Geographic scope	find_service/categoryBag/KeyedReference
Service QoS	find_binding/tModelBag

Canonical tModels. This function returns the operation tModel of the getCreditScore operation.

7.3 The PS-WSDL/USQL to ebXML Mapping

In order to take advantage of the expressiveness of PS-WSDL and USQL, we propose a mapping from PS-WSDL to the ebXML registry/repository version 3. This mapping is similar to the UDDI registry mapping with some differences that lie in the different data models of the two types of registries. The basis for this mapping, which is depicted in Table VI, is a best practice for registering Web services and their associated entities in an ebXML registry which is presented in Chiusano and Najmi [2003]. According to this best practice, a Web service can be represented in an ebXML registry through several registry information

```
<find_tModel generic="2.0" xmlns="urn:uddi-org:api_v2">
  <findQualifiers>
    <findQualifier>caseSensitiveMatch</findQualifier>
  </findQualifiers>
  <categoryBag>
    <keyedReference tModelKey="uuid:4A8DDFB0-7009-11DB-B81F-B0A4C6BCF510"
       keyValue="http://www.mybank.gr/LoansServicesSDO.owl#Customer_SSN" />
    <keyedReference tModelKey="uuid:795D0F00-7009-11DB-B81F-9C74A8AZC891"
       keyValue="http://www.mybank.gr/LoansServicesSDO.owl#Credit_Score" />
  </categoryBag>
</find_tModel>
```

Fig. 16. Example of UDDI inquiry function.

Table VI. Mapping of PS-WSDL to ebXML

PS-WSDL	ebXML Registry/Repository
Service	**Service**
Local name of service	Name
Documentation	description
Geographic scope	Collection of Location Classifications
Interface	
Domain	Collection of Domain Classifications
Endpoint	**ServiceBinding**
Address	accessURI
businessEntity	**Organization**
Local name of businessEntity	Name
Address and PhoneNum	PostalAddress and Telephone Number
URL	ProviderURL externalLink
QoSMetrics	**Collection of Slots**
Price	Price Slot
Availability	Availability Slot
Reliability	Reliability Slot

model classes, namely Service, ServiceBinding, and SpecificationLink [ebXML RIM 2005].

The additions that we propose in relation to the mappings described in Chiusano and Najmi [2003] appear in Table VI with gray background. In this mapping, the slot class is used in order to add QoS attributes to ServiceBinding instances through the specification of name/value pairs. The ability to dynamically add attributes to RegistryObject instances enables extensibility within the registry information model. However, the use of registry implementation-specific slots (such as QoS attributes) may not be interoperable across registry implementations and must be ignored silently by a registry that does not support such slots.

USQL queries take the form of an AdhocQueryRequest in the ebXML registry/repository. The AdhocQueryRequest contains a subelement (AdhocQuery) that specifies a query in one of the query syntaxes supported by the registry. ebXML allows filter-based and SQL-based search queries. A filter-based query is an XML syntax that provides simple query capabilities for the registry. SQL-based queries allow a client to submit complex SQL queries using a declarative query language. Figure 17 shows the XML representation of

```
<SubmitObjectsRequest>
  <rim:RegistryObjectList>
    <rim:AdhocQuery id="${QUERY_ID}">
      <rim:QueryExpression queryLanguage="${SQL_QUERY_LANG_ID}">
        SELECT * from Service s, Name_ nm
        WHERE (nm.parent = s.id AND UPPER(nm.value) LIKE UPPER("ConsumerLoans"))
        AND (s.id IN ( SELECT classifiedObject FROM Classification
        WHERE classificationNode IN ( SELECT id
        FROM ClassificationNode
        WHERE path LIKE
        "/urn:freebxml:registry:demo:schemes:iso-ch:3166:1999/Europe/GR")))
      </rim:QueryExpression>
  </rim:AdhocQuery>
</rim:RegistryObjectList>
```

Fig. 17. Example of query submission in the ebXML registry/repository.

an SQL query submission to ebXML registry/repository. This SQL query is the translation in the ebXML registry/repository of a USQL query that searches for Web services named “ConsumerLoans” and the geographic scope of which is Greece.

7.4 Discussion on Mediators

PYRAMID-S requires one mediator per each external registry type participating in the framework. Thus, the introduction of a new type of service registry necessitates the development of the appropriate mediator that implements the predefined mediator Web service interface. From a functional point of view, this roughly means that the new mediator should be able to receive PS-WSDL advertisements (or USQL queries), process and forward them to the new registry, receive the results from the latter, and send back the results to the requester. Mediator developers should also consider performance issues, as a nonefficient mediator may downgrade the PYRAMID-S performance. From a practical perspective, due to the Web service access of mediators, integration of a new mediator in PYRAMID-S is as simple as setting a couple of parameter values (such as registry type supported by the mediator and mediator’s access point) on the gateways that are going to use the new mediator(s).

We have to note that not all registry types support the whole of the PS-WSDL or USQL information. In such cases, it is clear that only the supported information is captured in these registries unless, whenever applicable, registry extensions are defined to support the new information. In our work, we have used the extensibility mechanism of UDDI in order to take advantage of semantic and QoS information in service advertisements and queries. Similarly, in the ebXML registry/repository, we have used the slot mechanism in order to add QoS support in service publication and discovery.

Another important feature that the developer of a mediator may decide to implement is indirect support for reasoning by taking into account the hierarchical relationships defined in the Standard Domain Ontology (SDO) or in the Domain Classification Ontology (DCO) and the semantic operators specified in the query. For example, consider the case where the user looks for all services

that belong to the "Banking-and-Investment" category of UNSPSC [UNSPSC] and to its subcategories (*extension* semantic operator in USQL). A reasoning-enabled mediator will identify the subcategories of "Banking-and-Investment" category through the DCO and it will build the query that it will forward to the registry in such a way that not only the "Banking-and-Investment" services will be retrieved but also the services belonging to its subcategories.[5] All in all, PYRAMID-S provides the requester with two options: (i) instruct the system to use the reasoning that may have been incorporated in the PYRAMID-S mediators or (ii) instruct the system to let this task to the registries. In the latter case, reasoning is performed only by the participating registries that have implemented it in their discovery algorithms.

8. PYRAMID-S EVALUATION

To demonstrate the viability of our approach, we have implemented a prototype system based on the aforementioned design. In this Section, we proceed with the description and the evaluation of this prototype.

8.1 PYRAMID-S Prototype Implementation

Our prototype system is based on the aforementioned design and includes mediators for the two most prominent registry types, namely UDDI and the ebXML registry/repository. The mediators have been developed according to the extensibility mechanism of PYRAMID-S and the mappings defined in Sections 7.2 and 7.3. Furthermore, the UDDI mediator that has been developed in the scope of this prototype implements indirect support for reasoning through the mechanism described at the end of Section 7.4.

The p2p infrastructure is mainly implemented by the four classes filled with light gray color in the class diagram of Figure 18. The *P2PNode* implements the functionality of simple peers (which is searching and performing actions in peer groups). The *ReplicatedIndexNode* extends the *P2PNode* by additionally implementing the searching service and the replicated index hosting and synchronization. The peer index of every *ReplicatedIndexNode* contains a *NodeDesc* object for the description of each peer node in the network. *Action* is the base abstract class for defining application-specific actions to be performed by peer groups (like *AddRegistryAction* and *AddDCODomainAction*). In PYRAMID-S it was decided that the *router* peers are *ReplicatedIndexNode* peers, due to their placement in the core of the architectural view of PYRAMID-S. *gateway* peers that may be considered slightly more peripheral are simple *P2PNode* peers.

The main part of the PYRAMID-S prototype (most of the classes and Web services) was implemented using Visual Studio .NET 2003 and the C# language. Only the ebXML mediator (class filled with dark gray in Figure 18) was developed in Java with NetBeans as the ebXML registry/repository implementation

[5]We would like to note that we have implemented such a reasoning-enabled mediator for UDDI which will be evaluated in the next Section, along with the rest of PYRAMID-S components.

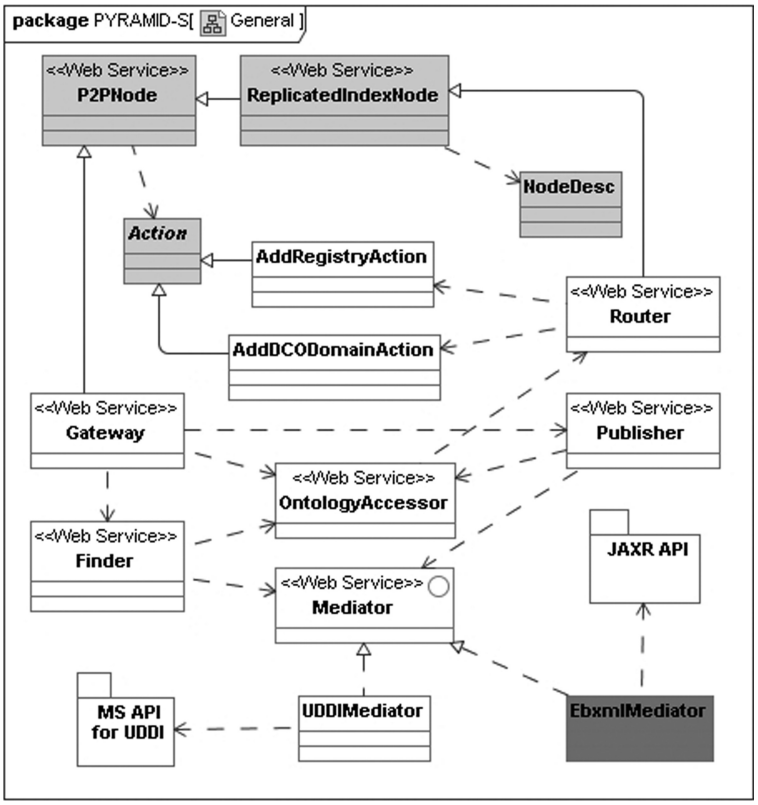

Fig. 18. PYRAMID-S class diagram.

that we used (OMAR) can only be accessed via JAXR API (JAVA API for XML registries).

The open-source Java implementation from the freebXML.org (called OMAR) was used for creating private ebXML registries. OMAR implements all the required features and nearly all optional features defined in the OASIS ebXML registry specifications version 3.0. The jUDDI open-source Java implementation of UDDI version 2 specification was used for creating private UDDI registries.

8.2 Experiment Setup

To assess and illustrate the feasibility of our approach, we have set up a PYRAMID-S environment for conducting service publications and queries. For the purposes of the experiment, we have built a testbed of 250 service advertisements (with around 1000 Web service operations) and 60 service queries, belonging to the domains of Financial-and-Insurance-Services and Travel-Food-Lodging-and-Entertainment-Services [UNSPSC]. The service advertisements (WSDL files) were retrieved by Web service catalogs [SEEKDA; WSLIST] and

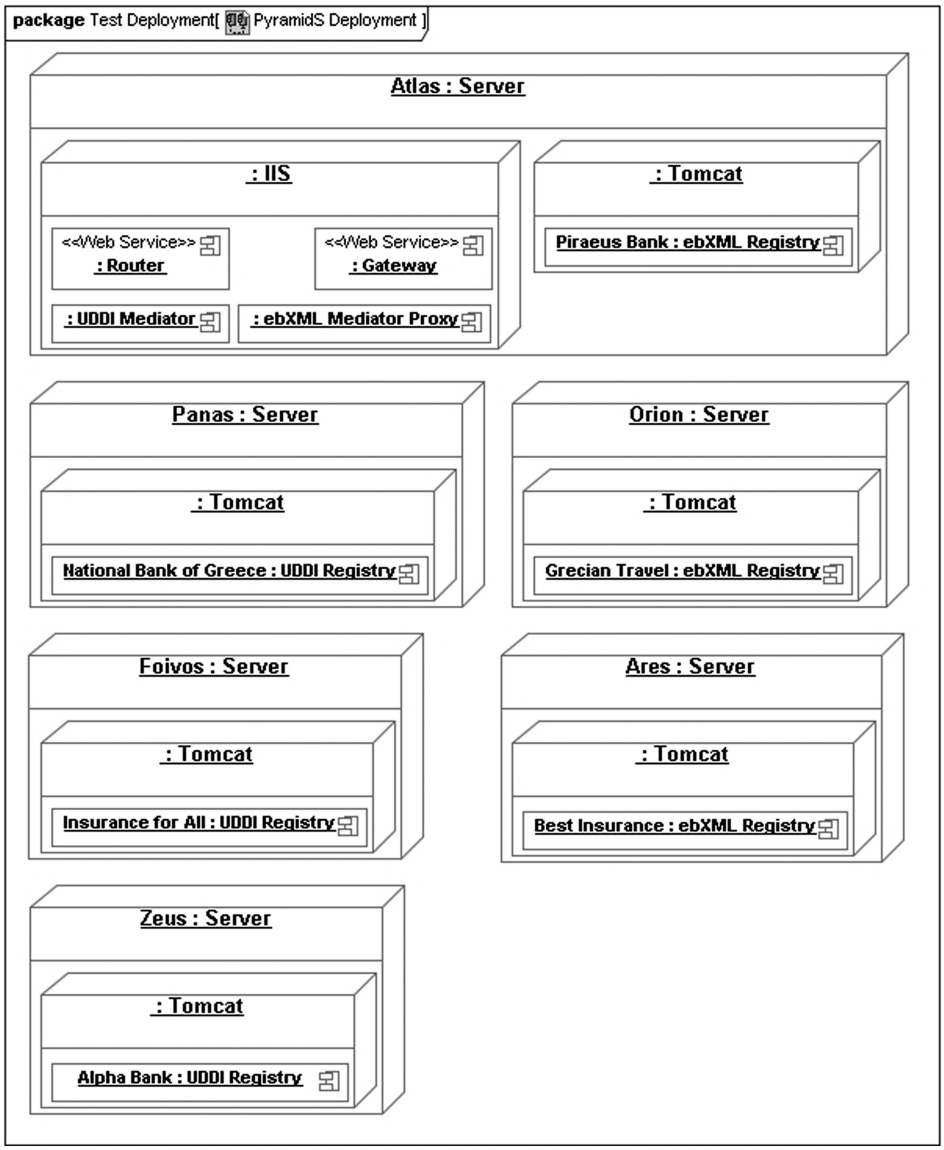

Fig. 19. Deployment diagram for PYRAMID-S experiment setup.

then they were annotated and published through the PYRAMID-S infrastructure, thus resulting in the corresponding PS-WSDL files. Similarly, the USQL files corresponding to the service queries were created. Service advertisements are expressed in PS-WSDL, and service queries in USQL. The environment consists of three UDDI registries and three ebXML registries, as can be seen in Figure 19.

The experiments were run on six machines with the following characteristics:

—Atlas: AMD Athlon 64 3400+, 1024 RAM, Microsoft Windows XP Home Edition;
—Panas: Intel Pentium 2399MHZ, 512 RAM, Microsoft Windows 2000 Professional;
—Orion: Intel Pentium 2399MHZ, 512 RAM, Microsoft Windows 2000 Professional;
—Foivos: Intel Pentium 2399MHZ, 1024 RAM, Microsoft Windows 2000 Server;
—Ares: Intel Pentium 2399MHZ, 2048 RAM, Microsoft Windows 2000 Server;
—Zeus: Intel Pentium 2399MHZ, 1024 RAM, Microsoft Windows 2000 Server.

Furthermore, as can be seen in Figure 19, one of these machines, Atlas, also runs the gateway and the router peers. As far as the network configuration is concerned, all the six machines were in the same local network.

8.3 Evaluation Results

Evaluation tests were conducted with a custom stress tool that we implemented for this purpose in Java. Since each request to a PYRAMID-S gateway is processed and further forwarded to one or more service registries, the stress tool generated traffic (service publication and discovery requests from our testbed of service advertisements and queries) at a user-defined rate and measured durations of executions.

Furthermore, due to the limited available hardware for hosting and labor for administering a large number of service registries, we decided to alter the code of several components, in order to simulate the behavior of PYRAMID-S in terms of execution time, as if it was communicating with a much larger number of UDDI and ebXML registries. We have to note that the execution time consists of four parts:

—the time for preparing the mappings to the appropriate registry format;
—the time for sending the requests;
—the execution time at the registries; and
—the time for handling the replies.

The time delays for sending actual requests and handling actual replies from service registries were measured during execution, multiplied by a given factor (50 for this experiment) and reintroduced as *wait* statements. The volume of data received from service registries (i.e., search results) was also multiplied by the same factor, so that further processing would be as time consuming as if the simulated service registries had replied in a way similar to the actual service registries. On the other hand, time for preparing the mapping was not reintroduced as delay, as the mapping is performed only once per mediator, regardless of the number of registries of this type. Similarly, execution times at the service registries were not reintroduced as delays, as service registries work in parallel in PYRAMID-S.

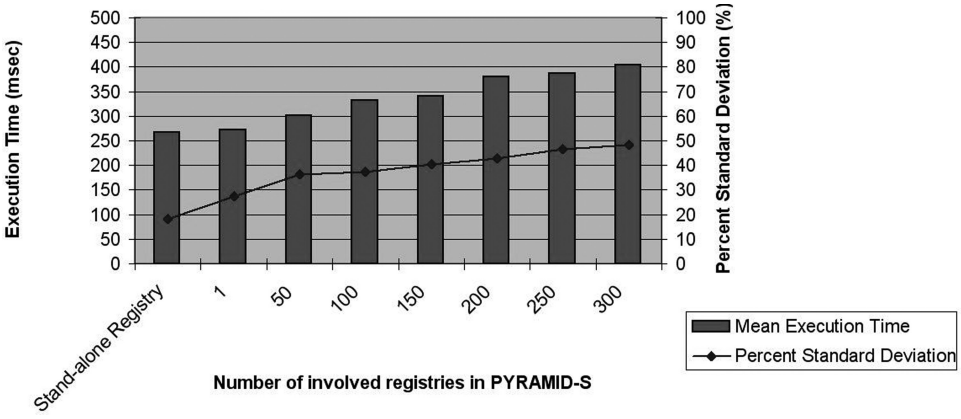

Fig. 20. Service publication execution time in PYRAMID-S.

The evaluation tests that we conducted revolved around the following axes.

—First, we aimed to *quantify the overhead* associated with the mechanisms introduced and prove that the advantages provided by PYRAMID-S framework do not come at the cost of performance. To this end, we have conducted comparative tests with direct publication and discovery on two stand-alone registries, namely UDDI and ebXML. These tests are indicative and intend to demonstrate the insignificant overhead introduced by PYRAMID-S.

—Second, we investigated the *scalability* of the proposed system, that is, PYRAMID-S scaling as the number of registries involved in publication/discovery requests increases. This was tested by increasing the number of registries where a request had to take place.

—Third, we examined the improvement in *precision* and *recall* that the use of semantic and QoS information brings in PYRAMID-S. For this reason, we have decided to perform comparative tests with a registry that does not support service capability and QoS information, such as UDDI.

As depicted in Figure 20 and Figure 21, service time difference between a stand-alone registry[6] and PYRAMID-S (with requests involving 1 registry) is insignificant, indicating a minimal PYRAMID-S overhead. Furthermore, the execution time of requests that involve 50, 100, 150, 200, 250, or 300 registries increases linearly, indicating that concurrency is effectively utilized (the registries are servicing requests in parallel). We have to note here that these execution times are indicative and may be different in case another UDDI implementation (instead of JUDDI) or ebXML implementation (instead of OMAR) are used. In any case, we have to make clear that the execution time in PYRAMID-S heavily depends on the execution time of the registries involved in the request (and more specifically it depends on the execution time of the slowest of the registries).

[6]Please note that the time depicted on the figures is the *average* value between the service time of a stand-alone UDDI registry and the service time of a stand-alone ebXML registry.

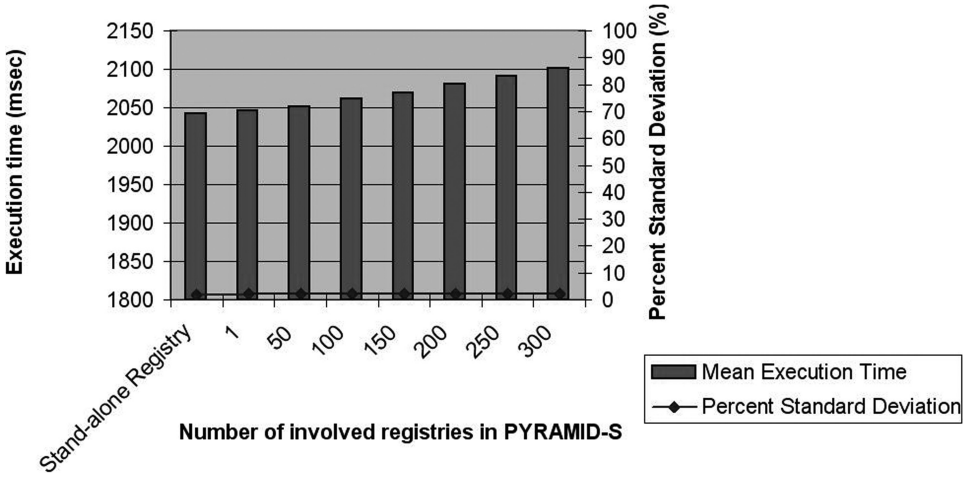

Fig. 21. Service discovery execution time in PYRAMID-S.

Focusing on scaling of performance in regard with the number of registries involved in each query, the experimental results suggest that PYRAMID-S scales well: As the number of registries involved in requests increases, execution time increases linearly with a very small rate. We have to note at this point that the scalability to large and very large domains may prove hard in case of complex ontologies. However, this problem also exists in a single stand-alone registry. In this work, we are not concerned with the improvement of the performance of semantic-enabled registries. Nevertheless, the PYRAMID-S framework can give a solution to this problem by splitting large domains to subdomains and by distributing services to registries associated with these subdomains.

In order to evaluate the improvement in precision and recall that PYRAMID-S brings, we have conducted a detailed experiment with PYRAMID-S and UDDI, using our testbed of service advertisements (corpus) and service queries. Our testbed contained 20 keyword-based queries where the requester's requirements are expressed as a set of keywords, 20 taxonomy-based queries, that is, queries for services belonging to a specific domain based on the DCO or geographic location based on ISO 3166-1999, 20 capability-based queries where the requester's requirements are expressed in the form of operation concept, service inputs, and outputs based on the corresponding SDO, and 20 capability- and QoS-based queries. For each of these queries we tried to obtain matching services from our entire corpus. In order to measure precision and recall, we had to know the set of all services that are relevant to a given query. Thus, for each query we hand-labeled the services in our collection that are relevant.

Figure 22 shows that keyword-based search yields coarse results with high precision errors both in UDDI and PYRAMID-S. More specifically, keyword-based search may return irrelevant services, which means that the requester needs to further inspect the retrieved services in order to find the relevant ones. The reason for this is that the query keywords might be syntactically equivalent

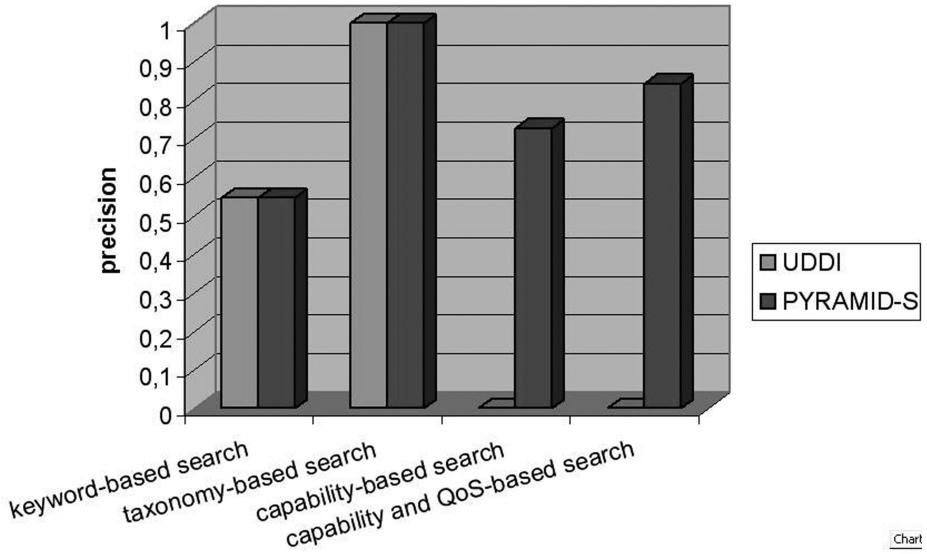

Fig. 22. Precision in UDDI and PYRAMID-S.

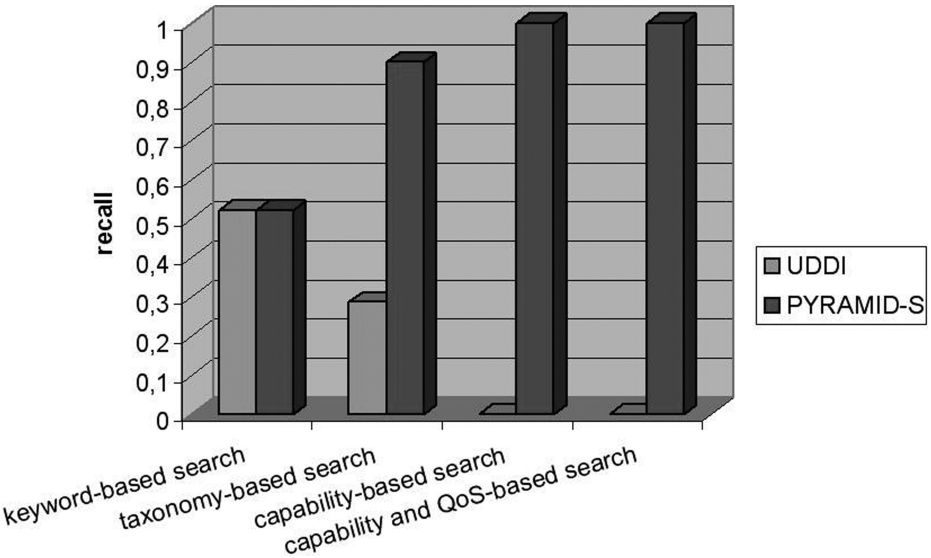

Fig. 23. Recall in UDDI and PYRAMID-S.

but semantically different from the terms in the service advertisement; for example, "commission" in the sense of piece of work and "commission" in the sense of payment to a bank for providing services (homonyms). Furthermore, the keyword-based search leads to low recall both in UDDI and PYRAMID-S, as depicted in Figure 23. A reason for this is that the query keywords might be semantically similar but syntactically different from the terms in service advertisements; for example, "loan" and "credit facility" (synonyms).

Regarding taxonomy-based search, the use of controlled vocabularies both in UDDI and PYRAMID-S assures 100% precision (Figure 22). However, the recall in UDDI is considerably lower than the recall of PYRAMID-S as UDDI does not support reasoning based on the supported taxonomies (Figure 23). The small percentage of missed services in PYRAMID-S (Figure 23) is due to the fact that the UNSPSC taxonomy provides many related domains, however, the publisher has to choose only one of them. So when a query is made on one domain, the results from other much related domains are not included, which leads to false negatives (relevant services not identified by PYRAMID-S).

Regarding capability-based search, UDDI does not support it at all. PYRAMID-S assures 100% recall through the use of controlled vocabularies and high precision. Precision errors may appear in several situations, such as:

—when there is no mapping from SDO to RDO for some concepts used in the query (in this case, less filtered results will be returned);
—when considering only the inputs and outputs for matching (this is not sufficient as inputs and outputs can be grouped differently to give different functionality; instead, operation names must be used as the matching unit, and these operations are in turn matched based on their inputs and outputs);
—when the service domain is not specified in the service request and the inputs and output concepts come from a general ontology, for example, time ontology; in such cases, the system will retrieve many irrelevant services.

All in all, the evaluation shows that the use of semantic and QoS information in PYRAMID-S provides high precision and recall by substantially improving on naïve keyword-based search. We have to note the following.

—The improvement in precision in comparison to UDDI is attributed to the fact that PYRAMID-S supports service capability and QoS information, whereas the recall is improved due to the fact that the UDDI mediator that we have implemented for PYRAMID-S performs reasoning on the semantic information. The comparison results would be similar for any other registry which supports neither service capability and QoS information nor reasoning.
—The use of QoS information in PYRAMID-S further improves the precision as it enables more pertinent service selection. In our experiments, this is demonstrated in the fourth scenario where a combination of capability and QoS criteria is used. However, it is evident that also keyword-based and taxonomy-based search can be benefited by the use of QoS information.
—PYRAMID-S achieves higher recall in case the registries involved in the request support reasoning or in case reasoning is implemented by the involved mediators. The more the involved registries or mediators that implement reasoning, the better the recall.

9. RELATED WORK

Over the last years, a lot of research has been carried out in the field of service publication and discovery, producing significant results. In the following, we review relevant research activities and compare them with our work.

In the area of service description, the initiative that mostly resembles PS-WSDL is SAWSDL [2006]. It started at the LSDIS Laboratory under the name WSDL-S [Akkiraju et al. 2005] and it is now maintained by a W3C working group. Similarly to PS-WSDL, SAWSDL takes advantage of the WSDL 2.0 extension mechanisms to build a simple and generic support for semantics in Web services. The extensions proposed by SAWSDL are limited to adding semantics to the *abstract* part of a service declaration. PYRAMID-S also extends the *implementation* part of WSDL by adding information about the geographic scope, the provider, and the QoS properties of the service. OWL-S [2006] and WSML [Lausen et al. 2005] are two other initiatives that aim at overcoming the shortcomings of WSDL. OWL-S is an OWL ontology for modeling various properties of a Web service. WSML is based on the WSMO [Roman et al. 2005] conceptual framework and constitutes a formal language for annotating Web services with semantic information. While both OWL-S and WSML are characterized by rich expressiveness, they suffer from considerable complexity. The PS-WSDL provided by PYRAMID-S is more lightweight and easier to apply, as it is relatively easy to update the existing tooling around the WSDL specification to accommodate the new information (i.e., semantics and QoS).

As far as the UDDI enrichment with semantic information is concerned, several approaches have been proposed. In Srinivasan et al. [2004], the authors define a mapping from OWL-S to UDDI and provide support for semantic matching by placing add-on modules on the registry side. This means that the existing UDDI infrastructure needs to be modified extensively to provide semantic support. Whereas this approach maps OWL-S service descriptions to UDDI, we preferred to work on an extension of WSDL as it is the commonly agreed standard. In Sivashanmugam et al. [2004], the authors propose a way of mapping WSDL-S to UDDI and then perform service discovery using query templates. As opposed to this approach, PYRAMID-S uses PS-WSDL for service description and USQL for service query. A comparison between PS-WSDL and WSDL-S (now SAWSDL) has been given before in the description of the improvements in service description. USQL is much more expressive compared to the query templates used in their approach.

Regarding the UDDI enrichment with QoS information, in Chen et al. [2004], QoS information about the services is kept in a component that stands between the user and the UDDI server. This same component sorts the service result according to the QoS metrics and then sends the result back to the requester. As opposed to this approach, PYRAMID-S stores QoS information in the UDDI registry. UDDIe [ShaikhAli et al. 2003] supports the storage of user-defined service properties (including QoS) in the UDDI registry and service discovery based on these. However, in contrast to our approach, this is achieved by modifying the existing UDDI specification. More specifically, they extend the businessService entity of UDDI with propertyBag and the find method in order to enable queries based on numeric and logical (AND/OR) ranges.

Regarding scalability in service publication and discovery, several solutions have been proposed such as UDDI version 3, PWSD [Li et al. 2004], SPiDeR [Syeda-Mahmood et al. 2005], METEOR-S [Verma et al. 2005] as well as some others [Papazoglou et al. 2003, Treiber and Dustdar 2007]. Some of them (e.g.,

UDDI version 3) use static configuration (which means that the links between cooperating registries are statically established) while others (e.g., PWSD [Li et al. 2004], SPiDeR [Syeda-Mahmood et al. 2005], METEOR-S [Verma et al. 2005], and the approach in Papazoglou et al. [2003]) use p2p technology. In the following paragraphs, we compare each of these approaches with PYRAMID-S.

The UDDI version 3.0 specification defines how registries may form a federation. A key aspect of such federations is the mechanism by which a registry entity may be promoted from one registry to another and how global registry key uniqueness is maintained. In this way, entities in a private registry, for instance, can be copied into another private registry for broader exposure or into a public registry for public consumption. The differences between this approach and ours are given in the following.

—PYRAMID-S allows data distribution based on domains (with the help of DCO) and the selection of the registries where a publication will take place is per publication. On the other hand, the data distribution in UDDI is governed by data management policies established among registries.

—In addition to data distribution, PYRAMID-S supports: (i) *structural heterogeneity* (differences in the data model), as it supports various types of registries through the use of mediators and (ii) *semantic heterogeneity* (i.e., semantic differences in the elements of the data model (content of the registries)) with the help of mediators and the mappings from SDOs to RDOs.

In Li et al. [2004], the authors propose a p2p-based Web service discovery architecture, called PWSD. Within the proposed framework, WSDL service descriptions are further annotated with metadata, provided by the service provider, and published on a specific peer, based on a hash key mechanism. Accordingly, XML-based queries are mapped to specific hash keys and thus forwarded to the appropriate peer hosting the desired service. However, within this system, service descriptions and service queries may only be expressed at the syntactic level, resulting in poor expressiveness. PYRAMID-S overcomes this drawback by using PS-WSDL and USQL support for semantic and QoS information.

In Syeda-Mahmood et al. [2005], a p2p-based Web service discovery framework, called SPiDeR, is proposed. In this framework, a subset of the participating service providers (those that have good resources such as high availability or high computing capacity) are dynamically assigned as superpeers and organized into a structured p2p system. Chord [Stoica et al. 2001] is used as the underlying overlay. These superpeers are responsible for indexing available services and resolving service lookups, thus they play the role of registries. SPiDeR supports three different types of search operations based on keywords, categories from a global service ontology, and service behavior. As opposed to this approach, PYRAMID-S also supports service advertisements and queries enhanced with capability information about the service by semantically annotating the service operations and their respective inputs and outputs. In SPiDeR, all superpeers have the same service indexing functionality. Thus, as opposed to our approach, SPiDeR does not support heterogeneous publication and discovery mechanisms.

In Verma et al. [2005], the authors introduce a p2p system, called METEOR-S, that is closely related to PYRAMID-S as it uses an ontology-based approach to organize registries, enabling domain-based semantic classification of WS. The main differences between the two approaches are the following.

—Both approaches use a peer-to-peer architecture, however, in PYRAMID-S the clients are excluded from the peer-to-peer network; the advantage of the latter is that service requesters and providers do not need to download and install any software. This results in zero deployment and maintenance cost.
—Both approaches use a p2p architecture, however, in PYRAMID-S the clients are excluded from the p2p network; the advantage of the latter is that service requesters and providers do not need to download and install any software. This results in zero deployment and maintenance cost.
—In METEOR-S the editing of the domain classification and registry information is performed only in a single peer, resulting in a single point of failure, while in PYRAMID-S editing is allowed in a number of peers, making provision for data consistency. This means that although the two approaches have similar registry lookup performance and scalability, PYRAMID-S is more fault tolerant.
—PYRAMID-S supports heterogeneous types of registries while METEOR-S necessitates that all participating registries comply with the UDDI specification.

Another approach to service publication and discovery is the one in Papazoglou et al. [2003], where an event-notification-based framework is proposed. This framework is based on the p2p technology and on the concept of service syndications, where related business form groups of interest. Each group has its own UDDI registry peer (called the superpeer) that stores a subdirectory of a UDDI business registry where every syndication peer publishes its service advertisement. The superpeer manages the communication between different peers and is responsible for the joining and leaving of peers of service syndications. The key concept of the service syndication is event notification, which allows a peer to subscribe itself for certain occurrences of events. The superpeer informs the registered peer when it obtains a matching publication from another peer. This enables peers to form their own so-called Peer Acquaintance Group (PAG), which consists of peers having the same interests and knowing each other. The members of the PAG cooperate by propagating WS requests to peers within their own PAG without the help of the superpeer. The differences between this approach and PYRAMID-S are the following.

—Both approaches use a p2p architecture, however, in PYRAMID-S the service providers and requesters are excluded from the p2p network; thus, it is not necessary to be part of the p2p network in order to publish or search for a service.
—PYRAMID-S supports heterogeneous types of registries, whereas this approach necessitates that all superpeers comply with the UDDI specification.

—All service providers in the marketplace (and syndication) use standard (unique) names for their port types and associated operations, thus semantic information is used in subscription/publication matching. However, PYRAMID-S goes one level deeper by also taking into account the inputs and outputs of the service operations.

In Treiber and Dustdar [2007], the authors propose a lightweight approach that uses the existing RSS infrastructure to construct an active distributed Web service registry. Every Web service provider offers a Web service registry news channel that serves as part of the distributed registry. Apart from the underlying infrastructure, the main differences between this approach and ours are the following.

—In their approach, the discovery process consists of either browsing through Web service registry content or subscribing to it to receive notifications about changes within the registry, whereas PYRAMID-S supports keyword-based and semantic search.

—Unlike PYRAMID-S, their approach does not support heterogeneous types of registries.

Concerning publication and discovery over heterogeneous registry types, the approach described in Baresi and Miraz [2006] is the only related work that we have found in the literature. In this work, the authors propose a framework, called DIRE, for the cooperation and federation of distributed and heterogeneous registries based on the publish/subscribe paradigm. DIRE aims at fostering the communication among different proprietary registries by means of two elements: a distributed *communication bus* and a *delivery manager* associated with each registry. The former interconnects the delivery managers and is based on a distributed publish and subscribe middleware. The latter acts as a façade: It is the intermediary between the registry and the bus and manages the information flow in the two directions. As opposed to our approach, DIRE disseminates services based on interests and requests, instead of according to semantic similarities among services (i.e., the domains that belong). In this way, DIRE supports a two-phase discovery process. The registry of an organization retrieves the services the organization is interested in and has subscribed to from the publish/subscribe infrastructure. The client always searches for the services it needs locally. Thus, the registry of an organization contains not only the services offered by the organization but also the services of other organizations to which this organization has subscribed. Furthermore, in DIRE, the client can express its requirements by using XPath expressions, which is mush less expressive than USQL.

As far as the matchmaking process is concerned, several algorithms have been proposed. In Stroulia and Wang [2005], the authors discuss a set of complementary methods for assessing the similarity of WSDL specifications, based on the semantics of the identifiers and the natural language descriptions as well as on the structure of their operations, messages, and types. In Dong et al. [2004], the authors describe the algorithms underlying the Woogle search engine for Web services. The proposed algorithms exploit the structure of the

Web services and employ a novel clustering mechanism that groups parameter names into meaningful concepts. The difference between these approaches and ours is that whereas these approaches support similarity-based search (linguistic similarity, term frequency), PYRAMID-S supports keyword-based and semantic search.

10. DISCUSSION

In this article we have presented PYRAMID-S, a framework that enables unified service publication and discovery over heterogeneous registry types. This framework has the following main characteristics.

—It categorizes and maintains services in domain specific registries.
—It supports unified Web service publication and discovery over heterogeneous registries, based on both syntactic and semantic information as well as QoS metrics.
—It has been designed and developed in a service-oriented way.

All these characteristics of PYRAMID-S entail the following advantages.

—Users are alleviated from the burden of handling the diversion between various technologies since they can uniformly publish or discover Web services. This is achieved by abstracting the interface of heterogeneous registries into a single unified view by the introduction of gateways as the entry point to PYRAMID-S.
—Higher recall, improved precision and better result ranking are enjoyed during service discovery by the use of syntactic, semantic, as well as QoS information about a service.
—It supports scalability by accommodating a large number of registries, routers, and gateways. Registry categorization is accomplished through a Domain Classification Ontology (DCO) helping in this way in narrowing the search context and improvement of performance. Overall system performance may be tuned by adapting the number of routers and according to business requirements.
—It is an open platform as it allows the integration of various types of WS registries, achieving thereby interoperation without affecting their specifications and autonomy.
—Last, the service-oriented design and implementation of PYRAMID-S allows the reuse of its modules for building other applications that need to perform service publication and/or discovery.

A prototype implementation of PYRAMID-S has been realized in order to assess the feasibility of the proposed solution and quantify the overheads associated with it. The preliminary evaluation results have shown that the proposed system scales well as the number of involved registries increases and that it provides the aforementioned advantages without incurring considerable overhead in service publication and discovery.

At this point, we would like to point out that the viability of our approach depends on two important issues.

—*Efficient Ontology Design and Management*: Designing and maintaining ontologies is a time-consuming activity as it requires agreement and consensus among domain experts. Thus, in the scope of PYRAMID-S several issues arise such as who is going to define the ontologies and who will check the validity of a new domain or the alteration of a domain in the Domain Classification Ontology (DCO).

—*Appropriate Regulation for the Involved Economic and Social Aspects*: There are several questions that need to be answered, such as: Who operates the routers and gateways? What benefits do they receive as a return on their costs in establishing and operating them? Surely, rewarding financially the operators of the gateways and the routers will result in better and more reliable services. Several solutions may be envisioned such as operators funded by their corresponding governments or by the various registries.

These issues are more easily addressed within a closed environment, such as in an organization or in a federation of organizations that is formed to serve a business domain or to accomplish a common goal. Further analysis of these issues is beyond the scope of this article.

Our immediate future plans are to perform further measurements in order to assess the time involved in each step of the publication and discovery process and to take advantage of the proliferation of SAWSDL and investigate the possibility to extend it in order to accommodate the information we have included in PS-WSDL, as it is not our intention to add yet another service description language and thereby increase the number of existing solutions.

REFERENCES

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Schmidt, M.-T., Sheth, A., and Verma, K. 2005. Web service semantics - WSDL-S, Tech. Note, version 1.0. http://lsdis.cs.uga.edu/ Projects/METEOR-S/WSDL-S.

Baresi, L. and Miraz, M. 2006. A distributed approach for the federation of heterogeneous registries. 2006. In *Proceedings of the 4th International Conference on Service-Oriented Computing (ICSOC'06)*. Springer, Berlin, 240–251.

Bernstein, P. A. and Melnik, S. 2007. Model management 2.0 - Manipulating richer mappings. In *Proceedings of the ACM SIGMOD International Conference on Management of Data.* ACM Press, 1–12.

Boag, S., Chamberlin, D., Fernandez, M. F., Florescu, D., Robie, J., and Simeon, J. 2006. XQuery 1.0: An XML query language.W3C Candidate Recommendation. http://www.w3.org/TR/xquery/.

Chen, Z., Liang-Tien, C., and Bu-Sung, L. 2004. QoS-Aware and federated enhancement for UDDI. *Int. J. Web Services Res. 1,* 2, 58–85.

Chiusano, J. and Najmi, F. 2003. Registering Web services in an ebXML registry, Version 1.0. Tech. note. http://www.oasis-open.org/committees/download.php/11907/regrep-webservices-tn-10.pdf.

Choi, N., Song, I., and Han, H. 2006. A survey on ontology mapping. *SIGMOD Rec.* 35, 3, 34–41.

Colgrave, J. and Januszewski, K. 2004. Using WSDL in a UDDI registry, Version 2.0.2. Tech. note. http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v202-20040631.htm.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. 2002. Learning to map between ontologies on the semantic Web. In *Proceedings of the 11th International Conference on World Wide Web*. ACM Press, 662–673.

DONG, X., HELEVY, A., MADHAVAN, J., NEMES, E., AND ZHANG, J. 2004. Similarity search for Web services. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB)*. VLDB Endowment, 372–383. ebXML RIM. 2005. ebXML registry information model (RIM) v3.0. http://www.oasis-open.org/specs/index.php#ebxmlrimv3.0.

EBXML RS. 2005. ebXML registry services specification (RS) v3.0. http://www.oasis-open.org/specs/index.php#ebxmlrimv3.0.

FENSEL, D. AND BUSSLER, C. 2002. The Web service modeling framework WSMF. *Electron. Commerce: Res. Appl.* 1, 2, 113–137.

GARCIA-MOLINA, H., HAMMER, J., IRELAND, K., PAPAKONSTANTINOU, Y., ULLMAN, J., AND WIDOM, J. 1995. Integrating and accessing heterogeneous information sources in TSIMMIS. In *Proceedings of the AAAI Symposium on Information Gathering*. 61–64.

GARCIA-MOLINA, H., PAPAKONSTANTINOU, Y., QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., VASSALOS, V., AND WIDOM, J. 1997. The TSIMMIS approach to mediation: Data models and languages. *J. Intell. Inform. Syst*.

GNUTELLA. 2002. Gnutella RFC. http://rfc-gnutella.sourceforge.net/.

LAUSEN, H., BRUIJN, J., POLLERES, A., AND FENSEL, D. 2005. WSML—A language framework for semantic Web services. In *Proceedings of the W3C Workshop on Rule Languages for Interoperability*. http://www.w3.org/2004/12/rules-ws/accepted.

LI, Y., ZOU, F., WU, Z., AND MA, F. 2004. PWSD: A scalable Web service discovery architecture based on peer- to-peer overlay network. In *Proceedings of the 6th Asia Pacific Web Conference (APWeb04)*. Springer, Berlin, 291–300.

MADHAVAN, J., BERNSTEIN, P. A., DOMINGOS, P., AND HALEVY, A. 2002. Representing and reasoning about mappings between domain models. In *Proceedings of the AAAI 18th National Conference on Artificial Intelligence*.

NAICS. 2007. North American industry classification system (NAICS). http://www.census.gov/epcd/www/naics.html.

OWL-S. 2006. Ontology Web language of services. http://www.ai.sri.com/daml/services/owl-s/1.2/.

PAPAZOGLOU, M. P., KRAMER, B. J., AND YANG, J. 2003. Leveraging Web services and peer-to-peer networks. In *Proceedings of the Conference on Advanced Information Systems Engineering (CAiSE'03)*. Springer, Berlin, 485–501.

PILIOURA, T., KAPOS, G.-D., AND TSALGATIDOU, A. 2004. PYRAMID-S: A scalable infrastructure for semantic Web service publication and discovery. In *Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications (RIDE'04)*. IEEE Computer Society, 15–22.

PILIOURA, T., TSALGATIDOU, A., AND KAPOS, G. D. 2006. Specification of PS-WSDL. Tech. rep. http://www.di.uoa.gr/~thomi/TR/PSWSDL.pdf.

PRUD'HOMMEAUX, E. AND SEABORNE, A. 2006. SPARQL query language for RDF.W3C Candidate Recommendation. http://www.w3.org/TR/rdf-sparql-query/.

ROMAN, D., KELLER, U., LAUSEN, H., BRUIJN, J., LARA, R., STOLLBERG, M., POLLERES, A., FEIER, C., BUSSLER, C., AND FENSEL, D. 2005. Web service modeling ontology. *Appl. Ontology* 1, 1, 77–106.

SAWSDL. 2006. Semantic annotations for Web services description language. http://www.w3.org/2002/ws/sawsdl/.

SEEKDA. Seekda homepage. http://www.seekda.com.

SHAIKHALI, A., RANA, O., AL-ALI, R., AND WALKER, D. W. 2003. UDDIe: An extended registry for Web services. In *Proceedings of the Service Oriented Computing: Models, Architectures and Applications (SAINT'03)*. IEEE Computer Society, 85–89.

SRINIVASAN, N., PAOLUCCI, M., AND SYCARA, K. 2004. An efficient algorithm for OWL-S-based semantic search in UDDI. In *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04)*. Springer, Berlin, 96–110.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the ACM SIGCOMM Data Communications Festival*. ACM Press, 149–160.

STROULIA, E. AND WANG, Y. 2005. Structural and semantic matching for sssessing Web service similarity. *Int. J. Coop. Inform. Syst. 14*, 4, 407–437.

SYEDA-MAHMOOD, T., SHAH, G., AKKIRAJU, R., IVAN, A., AND GOODWIN, R. 2005. Searching service repositories by combining semantic and ontological matching. In *Proceedings of the IEEE International Conference on Web Services (ICWS'05)*. IEEE Computer Society, 13–20.

TIAN, M., GRAMM, A., RITTER, H., AND SCHILLER, J. 2004. Efficient selection and monitoring of QoS-aware Web services with the WS-QoS framework. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, 152–158.

TREIBER, M. AND DUSTDAR, S. 2007. Active Web service registries. *IEEE Internet Comput.* 11, 5, 66–71.

TSALGATIDOU, A., PANTAZOGLOU, M., AND ATHANASOPOULOS, G. 2006. Specification of the unified service query language (USQL). Tech. rep. http://cgi.di.uoa.gr/∼michaelp/TR/usql-1.0-spec.pdf.

UBR. 2006. UBR shutdown FAQ. http://uddi.microsoft.com/about/FAQshutdown.htm.

UDDI. 2003. Universal description, discovery and integration v2 standard. http://www.oasis-open.org/specs/index.php#uddiv2.

UNSPSC. United Nations standard products and services code. http://www.unspsc.org/.

VERMA, K., SIVASHANMUGAM, K., SHETH, A., PATIL, A., OUNDHAKAR, S., AND MILLER, J. 2005. METEOR-S WSDI: A scalable infrastructure of registries for semantic publication and discovery of Web services. *J. Inform. Technol. Manag*. Special Issue on Universal Global Integration, 6, 1, 17–39.

WSLIST. Web service list. http://www.webservicelist.com.

WSDL. 2007. Web services description language. http://www.w3.org/TR/2007/WD-wsdl20-primer-20070326/.

Received January 2008; revised December 2008; accepted March 2009