METEOR–S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services

Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, Abhijit Patil,
Swapna Oundhakar, John Miller

Large Scale Distributed Information Systems (LSDIS) Lab
Department of Computer Science, University of Georgia
Athens, GA 30602
http://lsdis.cs.uga.edu
Email: {verma, kaart, amit, patil, swapna, jam}@cs.uga.edu

Abstract

Web services are the new paradigm for distributed computing. They have much to offer towards interoperability of applications and integration of large scale distributed systems. To make Web services accessible to users, service providers use Web service registries to publish them. Current infrastructure of registries requires replication of all Web service publications in all Universal Business Registries (UBR) which provide text and taxonomy based search capabilities. Large growth in number of Web services as well as the growth in the number of registries would make this replication impractical. In addition, the current Web service discovery mechanism is inefficient, as it does not support discovery based on the capability of the services and thus leading to a lot of irrelevant matches. Semantic discovery or matching of services is a promising approach to address this challenge. In this paper, we present a scalable, high performance environment for federated Web service publication and discovery among multiple registries. This work uses an ontology-based approach to organize registries, enabling semantic classification of all Web services based on domains. Each of these registries supports semantic publication of the Web services, which is used during discovery process. We have implemented two algorithms each for semantic publication and one algorithm for semantic discovery of Web services. We believe that the semantic approach suggested in this paper will significantly improve Web services publication and discovery involving a large number of registries. As a part of the METEOR-S project, we have leveraged the peer-to-peer networking as a scalable infrastructure for registries that can support automated and semi-automated Web service publication and discovery.

Keywords: Semantic Web services, Peer-to-Peer, Ontology, Semantic annotation of Web services, semantic Web services discovery, semantic Web services publication, domain-based registry, P2P UDDI

A number of new standards [1][2][3], tools [4], and applications have been developed recently to enhance the use of Web services. Significant progress has been made towards making Web services a pragmatic solution for distributed computing on the scale of the World Wide Web. However, there are a number of unresolved issues, which are hampering the wide scale deployment of Web services. One such issue is the need to improve the infrastructure for Web service discovery. We have investigated this issue as part of the ongoing METEOR-S project of the LSDIS Lab at the University of Georgia, which researches issues in Semantic Web Process Management by building upon techniques and technologies in workflow management, Web services and the Semantic Web. In this paper, we present METEOR-S Web Services Discovery Infrastructure (MWSDI), a scalable infrastructure for semantic publication and discovery of Web services.

At present, Web services are advertised in registries. The initial focus of Universal Description, Discovery and Integration (UDDI) specifications was geared towards working with a Universal Business Registry (UBR), which is a master directory for all publicly available Web services. However, the new version of the UDDI specification [5] recognizes the need for existence of multiple registries and the need for interactions among them. A large number of registry/repository implementations for electronic commerce, each focusing on registering services of interest to respective sponsoring groups, are also anticipated [6]. Hence, the challenge of dealing with hundreds of registries (if not thousands) during service publication and discovery becomes critical. Searching for a particular Web service would be very difficult in an environment consisting of hundreds of registries. This search would involve locating the correct registry in the first place and then locating the appropriate service within that registry.

The current approach [7] to solve the first challenge of finding appropriate registries, involves searching in UBR for Web services which access those registries. Searching for Web services in the private registries using this approach is inefficient as it involves first searching UBR for the relevant registry and then searching for relevant Web services in that registry. Finding the right services would be easier if the registries were categorized based on domains with each registry maintaining only the Web services pertaining to that domain. If the registries are specialized like this, search for services in that domain can be carried out in a relevant registry. For example, if a registry is related to the *Travel* domain, it will only maintain Web services specific to the *Travel* domain and search queries for Web services in *Travel* domain can be directed to it. In addition, adding semantics to the domain-registry association will help in efficiently locating the right registries based on discovery requirements. In MWSDI, we use a specialized ontology [1][8] called the *Registries ontology*, which maintains relationships between all domains in MWSDI, and associates registries to them.

The second challenge is that of finding the most-appropriate Web service within a registry. This challenge arises due to the discovery mechanism supported by UDDI. In an attempt to disassociate itself from any particular Web service description format, UDDI specification does not support registering the information from the service descriptions in the registry. Hence the effectiveness of UDDI is limited, even though it provides a very powerful interface for keyword and taxonomy based searching. Suggestions [9] have been made to register WSDL descriptions, which are the current industry accepted standard, in UDDI.. However, since WSDL descriptions are purely syntactic, registering them would only provide syntactical information about the Web services. The problem with syntactic information, is that the

[1] Ontologies are shared vocabularies that define concepts in a domain along with their properties and relationships.

semantics implied by the information provider are not explicit, leading to possible misinterpretation by others. Improving Web service discovery requires explicating the semantics of both the service provider and the service requestor. Our approach of improving service discovery involves adding semantics to the Web service descriptions and then registering these descriptions in the registries. Adding semantics to Web service descriptions can be achieved by using ontologies that support shared vocabularies and domain models for use in the service description. Using domain specific ontologies, the semantics implied by structures in service descriptions, which are known only to the writer of the description (provider of web service), can be made explicit. While searching for Web services, relevant domain specific ontologies can be referred to, thus enabling semantic matching of services. MWSDI provides support for this kind of matching by relating both Web service descriptions and user requirements to ontologies.

MWSDI provides an infrastructure for accessing multiple registries. The registries may be provided by different registry operators[2]. Each registry operator may support their own domain specific ontologies for their registries. They may also want to offer their own version of semantic publication and matching algorithms. Along with that, each operator may also provide their own value added services for the registry users. Thus, autonomy of the registry operators becomes a critical issue for the success of an infrastructure like MWSDI. For the functioning of MWSDI, the ontologies have to be efficiently distributed to users for service discovery and publication. With the increase in number of registries, scalability also becomes a significant issue. The recent paradigm of peer-to-peer networks, which are characterized by properties like autonomy and scalability, meet our requirements. Since each peer is an independent entity, it can have different roles in the network. In MWSDI, we have defined various roles for different peers[3]. Significantly, each registry is maintained by a peer. This gives us the desired autonomy, as each of these peers can support different services and ontologies. The framework we have used for creating the network has a number of protocols for peer discovery and communication between peers. We have used them to implement peer interaction protocols, which allow users to easily find relevant registries and communicate directly with the peers maintaining them. This decentralized approach makes MWSDI scalable as the number of registries increase.

We have implemented the MWSDI specifications as a prototype system that allows different registries to register in a P2P network and categorize registries based on domains. These registries will in turn support domain specific ontology and provide value added services for performing registry operations. We have also implemented and tested two algorithms for Semantic publication and discovery of Web services as value added services for the registries. Using the MWSDI and these algorithms can significantly improve upon the current standards in Web service registration and discovery. With Web services being the enabling technology for achieving virtual enterprises, the success of inter-enterprise application interoperability will be limited by the discovery mechanism of Web services. With the growing trends like e-market places including e-services and e-utilities for domain specific services and exposure of enterprise services using semi-private registry implementations, we believe that an infrastructure like MWSDI will help organizations and businesses in carrying out their business goals in a more scalable environment.

[2] a company or organization that runs an instance of the publicly accessible Web service registry
[3] details of the different peer roles are given in section 2.3.

In this paper we describe the architecture, prototype implementation and working of MWSDI. The main contributions of this work are:

- Creating a scalable infrastructure for accessing multiple registries
- Semantically dividing registries into domains using semantics for improved service publication and discovery
- Implementing two approaches for annotating service descriptions (WSDL) and an algorithm for semantic publication of Web services in UDDI
- Implementing an algorithm that uses these semantics during service discovery

The rest of the paper is organized as follows: Section 1 briefly summarizes the background. Section 2 presents the architecture. The implementation details are discussed in Section 3. Section 4 gives a detailed description of semantic publication and discovery using our infrastructure. Section 5 lists the related works. Finally in Section 6, we outline our intentions for future work.

1. Background

This section details the background material relevant to this research. We cover peer-to-peer computing, Web services and related technologies and the Semantic Web, discussing state of the art and their relevance to METEOR-S in general and to this work in particular.

1.1 Peer-to-Peer (P2P) Computing

P2P computing is considered the next evolutionary step in the way computations are done. This new direction in distributed computing focuses on networking and resource sharing aiming at better reliability and scalability. There have been many attempts to define P2P networks [10]. Comparing P2P networks with client-server networks helps in defining them. In a client-server architecture, servers provide resources or services and clients use them. These roles are not reversible in this architecture. However, in P2P architecture, all the entities can act as provider or requester of resources or services. All these entities have interchangeable roles unlike the client-server architecture. Depending on the level of decentralization, P2P networks are classified as “pure” or “hybrid”. In a pure P2P network, all peers have equal roles and there is no centralization. However, in hybrid P2P networks, some resources or services are centralized. P2P networks scale well with increase in number of resources maintaining their autonomy.

MWSDI aims to provide unified access to a large number of registries, which may be maintained by different operators. As a result, a large degree of autonomy is required, implying that the infrastructure should be distributed. This infrastructure should also scale with the increase in number of registries. This kind of autonomy and scalability is provided by P2P networks.

1.2 Web Services

Web Services are described as reusable software components that interact in a loosely coupled environment [11]. The core components of the Web services infrastructure are XML based standards like WSDL, UDDI and SOAP. Web services description is done using

WSDL[3]. Like the name suggests, it is a language for describing the interface and protocol bindings of web services. “UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily, and dynamically find and use Web services over the Internet” [1]. Simple Object Access Protocol (SOAP) is the standard message protocol for Web services. “It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses”[2].

Due to the fact that Web services are based on XML standards, they are currently being used by enterprises for interoperability. As a result, companies convert their applications to Web services to make disparate applications interact. Apart from that, companies may have number of Web services specifically for their partners and other Web services for public use. A lot of companies may prefer operating their own registries leading to a number of private implementations. However, the companies may want their registries to be found by their business partners and other entities. These companies may also want to expose their workflow repositories as their services registry. The current solution is publishing their registries as Web services in the UBR. As the number of registries increase, searching for a Web service would add the overhead of finding the relevant registry. MWSDI approaches this problem by providing a unified view of all the registries meaning that the companies may use this infrastructure to abstract the details of their registry implementations thereby providing simple and common means of accessing them.

1.3 Semantic Web

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [12]. Use of ontologies to provide underpinning for information sharing and semantic interoperability has been long realized [13], [14], [15]. In the Semantic Web, we not only have an opportunity to add semantics to information resources like Web pages, but also to Web Services, enabling sharing and integration of information resources as well as applications [16]. These shareable definitions called semantic annotations [17], utilize ontologies. In MWSDI architecture, ontology-enabled semantics is used for two purposes: dividing registries into domains, and semantic annotation of Web services.

2. Architecture

The layered architecture of MWSDI is discussed in this section. MWSDI is divided into four different layers, namely the Data layer, the Communications layer and the Operator Services layer and the Semantic Specifications layer. The layered architecture is shown in Figure 1. The Data layer is comprised of the Web service registries that are part of MWSDI. The Communications layer allows all the different components to communicate with each other. The Operator Services layer enables registry operators to support various kinds of services that operate on their registries. The Semantic Specifications Layer is orthogonal to all these layers, as it spreads across all the layers. Following subsections explain each of these layers in detail.

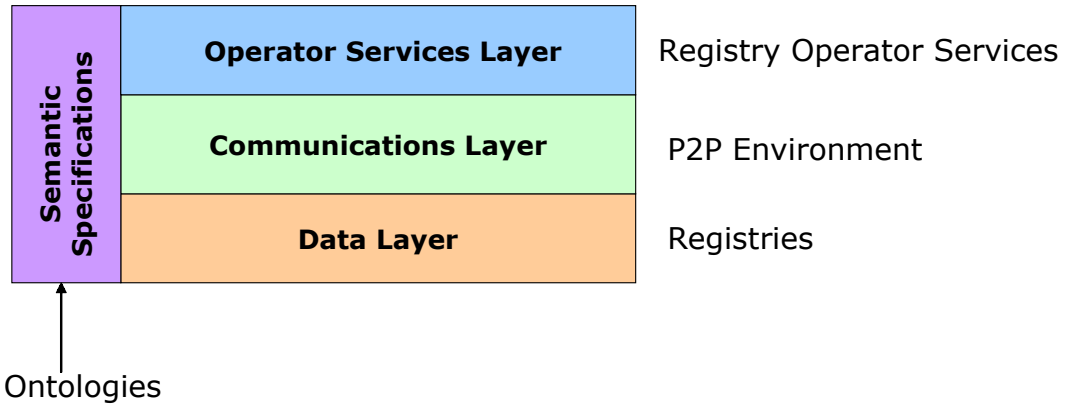


Figure 1: Layered Architecture of MWSDI

2.1 The Data Layer

The Data layer consists of the Web services registries in MWSDI. Since UDDI is considered a standard for web services registries, we have used only UDDI registries in our implementation and testing. To remain consistent with UDDI specifications, we have not made any changes to the way the registries are accessed. The registries can therefore be accessed in a standalone manner. However for semantic publishing and discovery of Web services, the Operator Services layer needs to be accessed

2.2 The Semantic Specifications Layer

The role of the Semantic Specifications layer is to enable the use of semantic metadata. We add semantics at two levels in MWSDI, *i.e.* at the level of the registries and at the level of individual Web services in each registry by using ontologies. We have used the Protégé [18] API to create, store and manipulate the ontologies.

2.2.1 Semantics at the Registries Level

At the level of registries, we have a specialized ontology called the *Registries Ontology*. This ontology maps each registry to a specific domain thereby grouping them based on domains. In addition, it stores properties of registries, relationships among Registries and relationships among domains. Properties of each registry may include the registry specification name, the registry specification version, the API supported, the registry operator details, quality of service (QoS) of the registry, access URLs and the constraints in accessing that registry. The relationships that are captured in the *Registries Ontology* could be the kind of affiliation between different registries or the relationships between different domains. Figure 2 shows the sample structure of the *Registries Ontology*. The use of *Registries Ontology* is explained in the following sections.

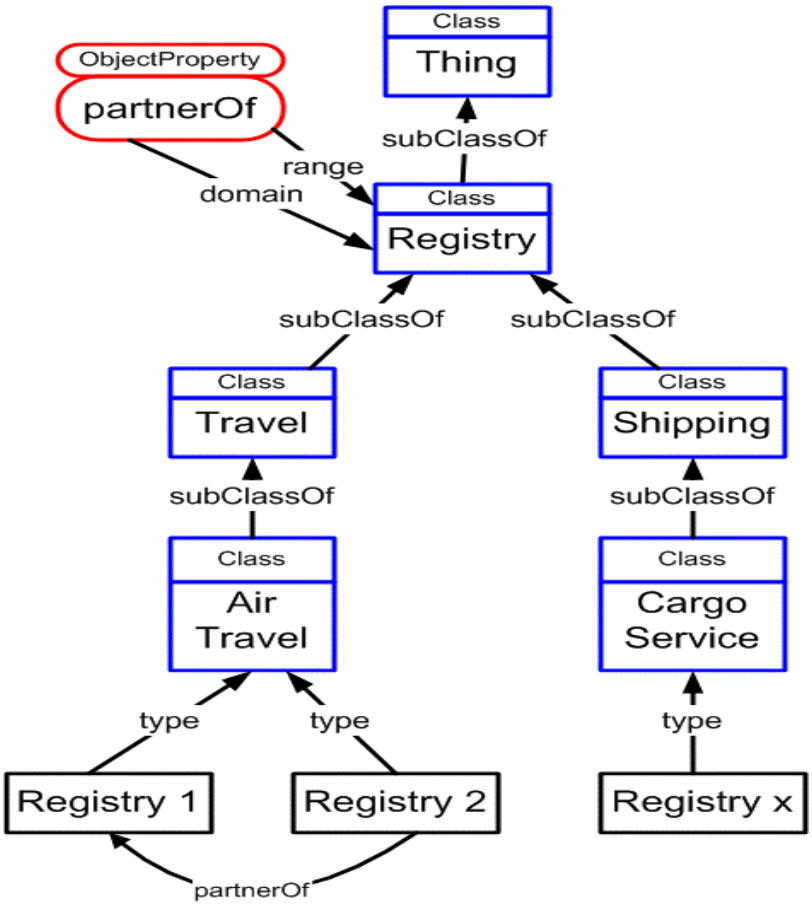


Figure 2: Sample *Registries Ontology*

2.2.1.1 Mapping between Registries and Domains

As mentioned earlier, every registry in MWSDI should be mapped to a specific node in the *Registries Ontology*. This node is typically a domain that represents the functional domain of services in the registry. A registry is also allowed to be mapped to multiple domains. The mappings between registries and domains are used to group the registries based on domains. In this way finding Web services in a specific domain could be limited to only the registries that are mapped to that domain. To search for a Web service in a certain domain, a user may either opt to select a particular registry in that domain or he may prefer searching in all the registries mapped to that domain. Every time a new registry operator joins the MWSDI infrastructure, the new registry has to be mapped to a domain. The details of the new registry, its mapping to domains and other relevant details are written into the *Registries Ontology*. This is called *Registries Ontology* update. If a new registry operator wants to map his registry to a domain that does not exist in the registries ontology, he is allowed to create that domain to map his registries with. Hence the registry operator upon joining the MWSDI infrastructure can either associate his registry to an existing domain in the ontology or he can update the ontology with a appropriate domain and associate his registry to that new domain. *Registries Ontology* update also establishes relationships among domains. For example, we may have Travel domain in the registries ontology that is associated with a group of registries. If an airline company wants to

run a semi private registry having the list of its end-user services (*ticketBooking*, *flightEnquiry* etc.) and other Business Interaction services (Inter-organizational workflow tasks, internal services like order processing, inventory control etc.), then it could map itself to nodes like *AirTravel*. Supposing that *AirTravel* doesn't exist he is allowed to create it as a new sub-domain under Travel and map his registry to the new domain. When a new domain is created it is related to an existing domain using any kind of relationships helping to establish relationship among domains. Mapping registries into domains allows us to route queries directly to relevant domains. Considering an example search for Web services that give prices of tickets from Atlanta to New York, the search could be carried out over all the registries in the *AirTravel* domain. A search query could also use domain-domain relationships. It could be a simple subsumption relationship or could be other type of named relationships. Suppose there is another Airline company, which at the time of joining MWSDI creates a domain called *AirLineServices* and then relates it to the domain *AirTravel* using "equivalentOf" relationship. A search query directed to either could be forwarded to the other registry. The current replication mechanism found in UBRs is also supported in MWSDI. These UBRs could be mapped to a node in *Registries Ontology* named 'Universal", which is a root domain of all other domains. These UBRs could also be related to each other using the relationship 'replicateOf'. In this way search for a web service need not be carried in all the registries but can be limited to any of these registries.

2.2.1.2 Registry – Registry Relationships

The *Registries Ontology* also captures the properties of all the registries and relationships among them. For example, consider two registries, *Registry1* and *Registry2*, which are run by two Airline companies. Each of these companies maintains Web service registries that list their Web services that are available for public access. *Registry1* could be from "Intercontinental Airlines" and the access URL could be "*http://www.ica.com/regUrl*". These details are stored as properties of the registry. *Registry2* could be from a partner company of "Intercontinental Airlines" and hence *Registry2* could be related to Registry1 using the relationship "partnerRegistryOf".

Considering the above mentioned example, search for Web services that give prices of tickets from Atlanta to New York, the number of registries searched can be made more selective using the relationships stored in the ontology. If the user wants to buy tickets only from Intercontinental Airlines and its affiliates, the relationship "partnerRegistryOf" could be used to execute the search only over registry *Registry1* and all other registries like *Registry2* which share this relation with *Registry1*.

2.2.2 Semantics at the Web Services Level

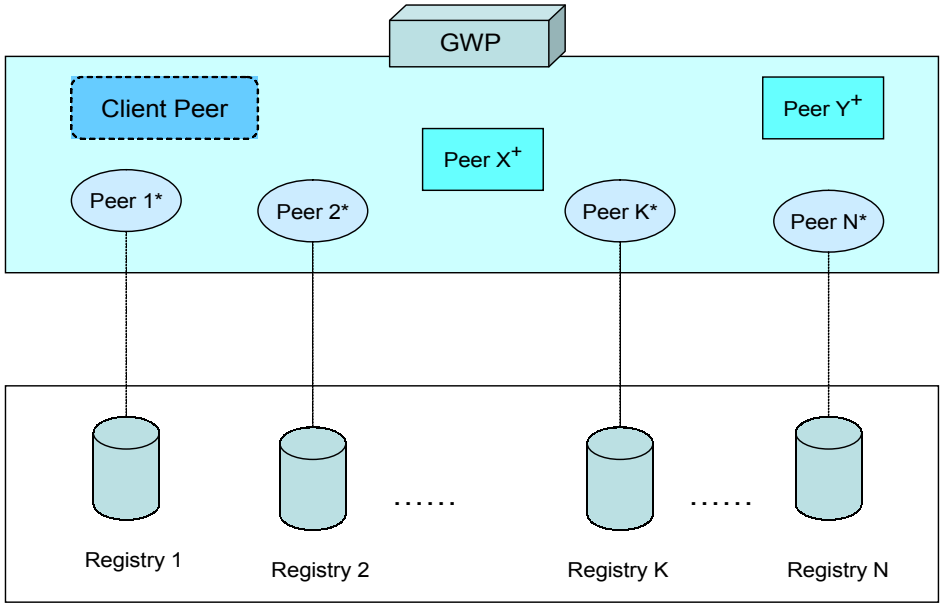
At the level of individual Web services, we have domain specific ontologies supported by each registry for semantic publication and discovery of the Web services. We envision registry operators creating their own domain specific ontologies[4]. The domain specific ontologies are created from concepts and terminologies that are likely to be used by Web services in a particular domain. For example, in the electronic commerce domain, the domain specific ontology can

[4] In this paper, we do not discuss how the domain specific ontologies are created. [21], [22]

consist of concepts from the ebxml Core Component Dictionary [19]. Another example could be a domain specific ontology for the *AirTravel* domain which may include concepts like *TicketPrice*, *AirportName, FlightNumber, ArrivalCity and DepartureCity.* We add semantics to Web services by mapping input and output types in their descriptions to concepts in the domain specific ontologies. These mappings can either be done semi-automatically or manually and stored in UDDI data structures. We have implemented both these approaches and provide them as two different Operator Services in the service layer. In the first Operator Service, we have adopted a method similar to the one presented in [20]. We manually map input and output types in the WSDL files and store the mappings in UDDI data structures. In the second service, we store semi-automatically perform these mappings and the details are stored in UDDI.

2.3 The Communications Layer

The Communications layer consists of a peer-to-peer network, which provides an infrastructure for the distributed components of MWSDI to communicate with each other. All the components in our network are implemented as peers.



GWP	Gateway Peer controls access to the peer-to-peer network for new registry operators
Peer 1* – Peer N*	Operator Peers run Operator Services and act as providers of *Registries Ontology*
Peer X^+, Peer Y^+	Auxiliary Peers only act as providers of the *Registries Ontology*
Registry 1 – Registry N	Web service registries

Figure 3: Components of the Communications Layer along with Registries

MWSDI has four different types of peers depending on their roles: the Operator Peer, the Gateway Peer, the Auxiliary Peer and the Client Peer. Figure 3 shows the different types of peers

in the Communications layer. Each Operator Peer maintains a registry (depicted by dotted lines in Figure 3). The role of the Operator Peer is to control a registry and to provide Operator Services for its registry. The Operator Peer also acts as a provider for the *Registries Ontology* to all other peers who need it. We discuss the Operator Services layer in the next section. The Gateway Peer acts as an entry point for registries to join MWSDI. It is responsible for updating the *Registries Ontology* when new registries join the network. It is also responsible for propagating any updates in the *Registries Ontology* to all the other peers. Gateway Peer is not associated with any registry.

Registries Ontology is important for semantic publishing and discovery. Making this ontology highly available is critical to the performance of the infrastructure. Hence, we have dedicated peers called Auxiliary Peers, which only act as providers of the *Registries Ontology*. The Client Peers are transient members of the peer-to-peer network, as they are instantiated only to allow users to use the capabilities of the MWSDI.

We classify the peer-to-peer network used by our network to be hybrid because the Gateway Peer is the only peer that can update the *Registries Ontology* or initiate new peers. While the Gateway Peer is a single point of failure for ontology updates, it does not impact discovery and publishing of Web services, as they are provided by other peers. In case of failure of the Gateway Peer, only initiation of new registries will not be possible. We have implemented recovery mechanisms for restarting the Gateway Peer. The peer interactions are discussed in detail in sections 3.1 and 3.2.

2.4 The Operator Services Layer

The Operator Services layer maintains all the services provided by the Operator Peers that operate on their registries. Operator Services are the value added services like semantic discovery and publication of Web services, provided by the registry operators. This layer also has a special service using which domain specific ontologies can be downloaded at the client end. Using these services, this layer abstracts users from intricate details in using semantics for Web service publication and discovery in the registries. The Client Peers communicate with the registries using this layer. The users can select relevant registries using the Client Peer's user interface and create templates[5] for discovery or publishing in these registries. These templates are communicated to the Operator Services layer, which translates the templates to the registry specific format and performs desired function. Different registry operators could provide different algorithms for Semantic publication and discovery. The internal workings of these algorithms are abstracted from the user, as the user just has to create the templates and send it to the relevant Operator Peer and invoke the desired Operator service.

This layer can also be used to deploy services for various tasks like Web service composition [23]. Registry operators can also provide value added services according to the domain or functionality. We have implemented two services for semantic publication and one service for semantic discovery of Web services in UDDI Registries. Conventional UDDI querying based on keyword matching is also supported as services of the Operator Services layer.

[5] details of the template creation are given in section 4

In the case of a non-UDDI registry implementation, the registry provider can use the Operator Services layer to provide the needed abstraction thereby supporting SOAP based access to that registry. This layer can provide a wrapper service that can be used to translate the registry entry details to UDDI data structure specifications and vice versa during the SOAP message processing. Even if this kind of translation is not supported, capturing the specification of the registry and the details of the relevant access API in *Registries Ontology* will help in registry selection.

3. MWSDI Implementation

MWSDI architecture has been implemented on a cluster of SUN workstations as peer-to-peer network using the JXTA framework [24]. Any peer can be a JXTA peer if it implements one or more JXTA protocols. While there are a number of such protocols, we have used the Peer Discovery Protocol and the Pipe Binding Protocol. The Peer Discovery Protocol enables a peer to find other peers. Pipes are communication channels in JXTA networks. They are virtual entities, implying that their endpoints can be bound to more than one peer. The Pipe Binding Protocol is used to bind a pipe to a peer at runtime. In addition to these, we have implemented a number of peer interaction protocols and peer roles to meet our requirements. The Peer Interaction Protocols we have implemented are the Operator Peer Initiation Protocol and the Client Peer Interaction Protocol. Our aim is to develop a scalable infrastructure of registries and this infrastructure should be universally accessible meaning that all devices like PDA, Cell phones, PCs etc should be able to get into the network of registries to make Web service discovery. There are already few PDA specific Web services available in the market. Hence with the use of JXTA which enables interoperability and platform independence, our infrastructure supports all kind of devices to do service discovery on a community of registries.

3.1. Operator Peer Initiation Protocol

The Operator Peer Initiation protocol defines the process involved in adding a new registry to the MWSDI system. Since mappings between all the Registries and their respective domains are maintained in the *Registries Ontology*, it must be updated every time a new registry is added. As the Gateway Peer is responsible for maintaining the consistency of the *Registries Ontology*, it is the only existing peer, which can be contacted by new registry operators to join MWSDI. The interaction diagram of the Operator Peer Initiation Protocol is shown in Figure 4.The process is initiated by a new peer. It joins the network and requests Gateway Peer for *Registries Ontology*. Any random peer, who acts as a provider of *Registries Ontology*, responds from the peer group. Using the *Registries Ontology*, the new peer can associate his registry either to an existing domain or a self-created domain. These details of the update are then sent to the Gateway Peer, which uses its locking and versioning mechanism to update its version of the ontology. The Gateway Peer then sends an acknowledgement to the new peer, which then joins the network as an Operator Peer. We have developed a concurrency control mechanism to allow the Gateway Peer to simultaneously initiate a number of new peers. The updated *Registries Ontology* is then communicated to the existing Operator Peers. In the future we plan to add security measures in this protocol during ontology update.

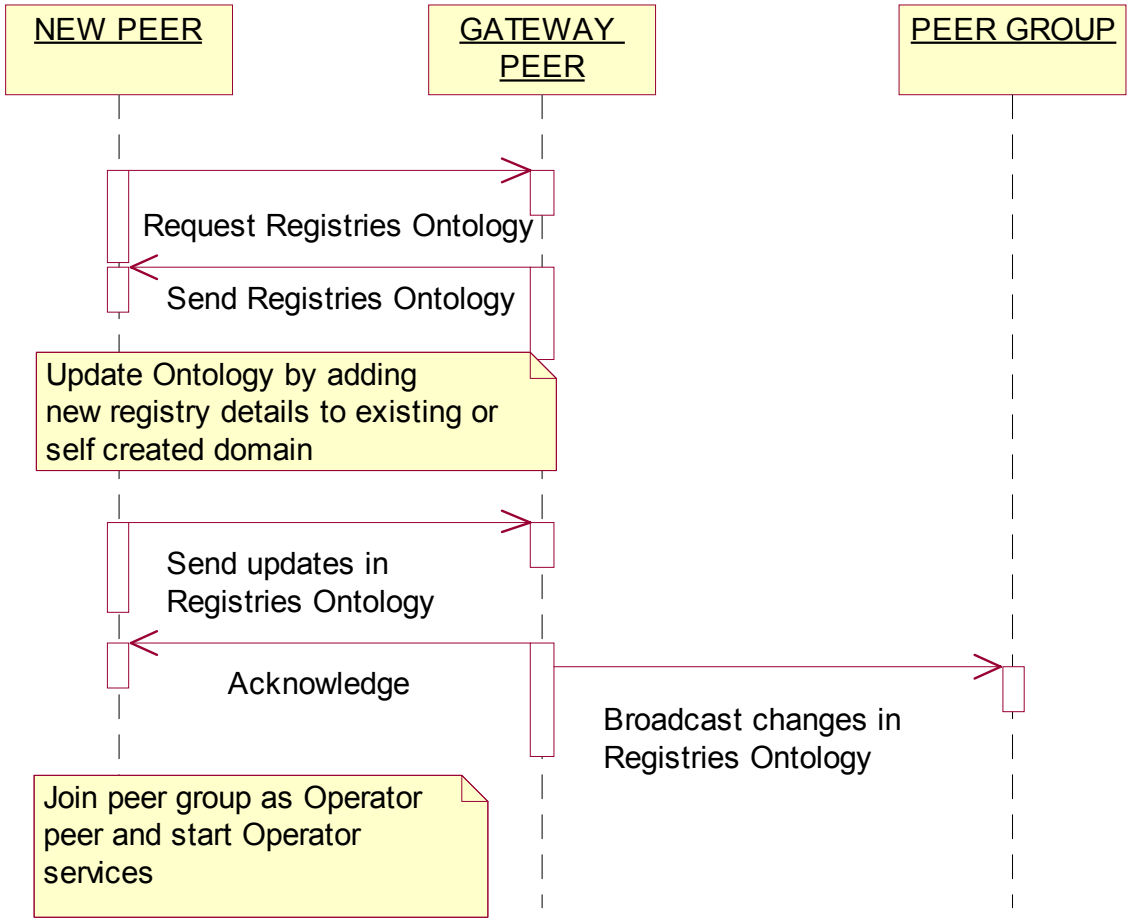


Figure 4: Interaction Diagram for Peer Initiation Protocol

3.2. Client Peer Interaction Protocol

In order for clients to access the Operator Services, we have implemented the Client Peer Interaction Protocol. Users need to download the Client Peer code and use it to access the MWSDI. This Client Peer enters the network as a transient peer and makes a request for the *Registries Ontology*. This request is answered by any peer in the network which acts as a provider for the *Registries Ontology*. The *Registries Ontology* is then displayed as a taxonomy of domains by the clients user interface and the users can use it to select a relevant domain for service publication and discovery. The client interface displaying *Registries Ontology* is shown in figure 5.

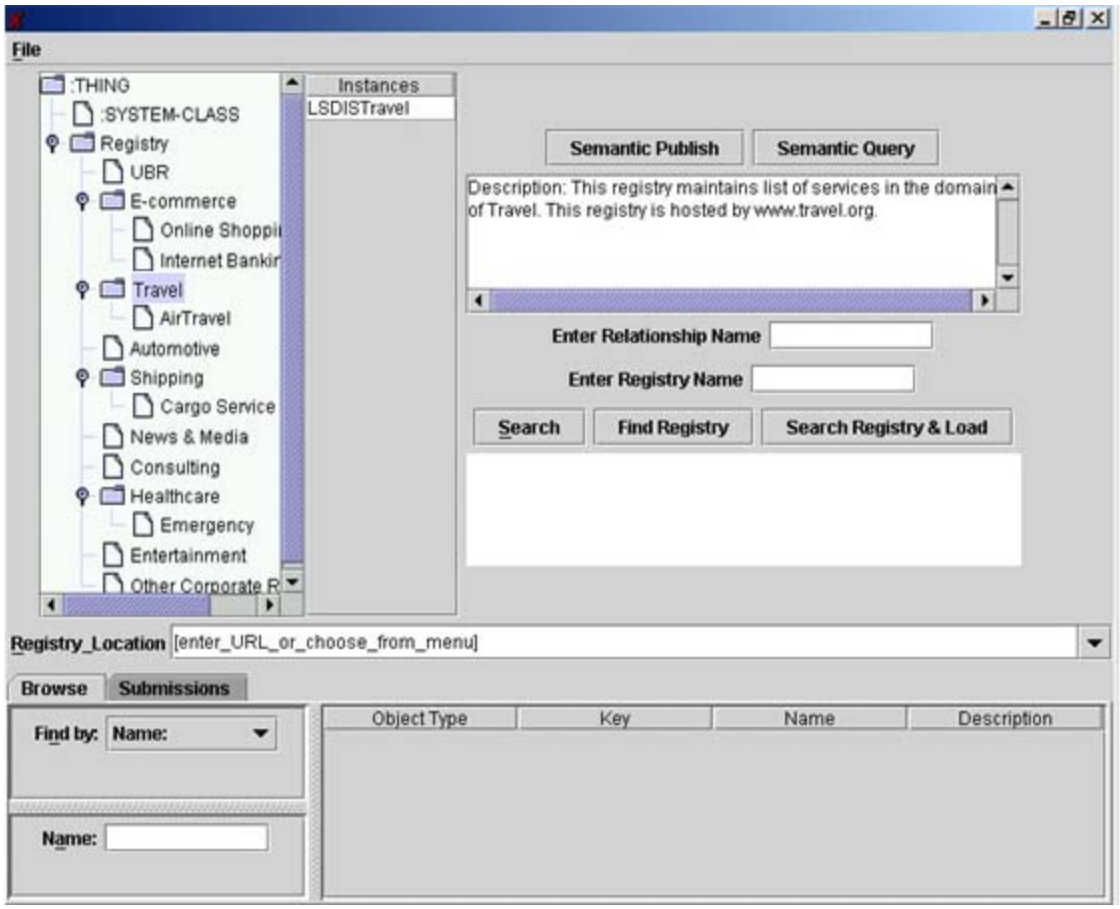


Figure 5: *Registries Ontology* displayed as a Taxonomy of Domains

Figure 6 shows the interaction diagram for Web service publication. After the user selects the domain in the *Registries Ontology*, all relevant Operator Peers in that domains are requested by the Client Peer for domain specific ontologies. Users can then choose the most relevant domain specific ontology and send their Web service publication details to the relevant Operator Peer, which executes the appropriate Operator Service to publish the Web service in the registry it maintains.

The Client Interaction Protocol for semantic discovery is almost the same. The user chooses the appropriate domain from the *Registries Ontology*. Next the Client Peer requests all Operator Peers in that domain for the domain specific ontologies. Then the user selects the most relevant domain specific ontology and sends the discovery details to the corresponding Operator Peer. The Operator Peer then executes the appropriate Operator Service to query the registry and returns the results to the Client Peer.

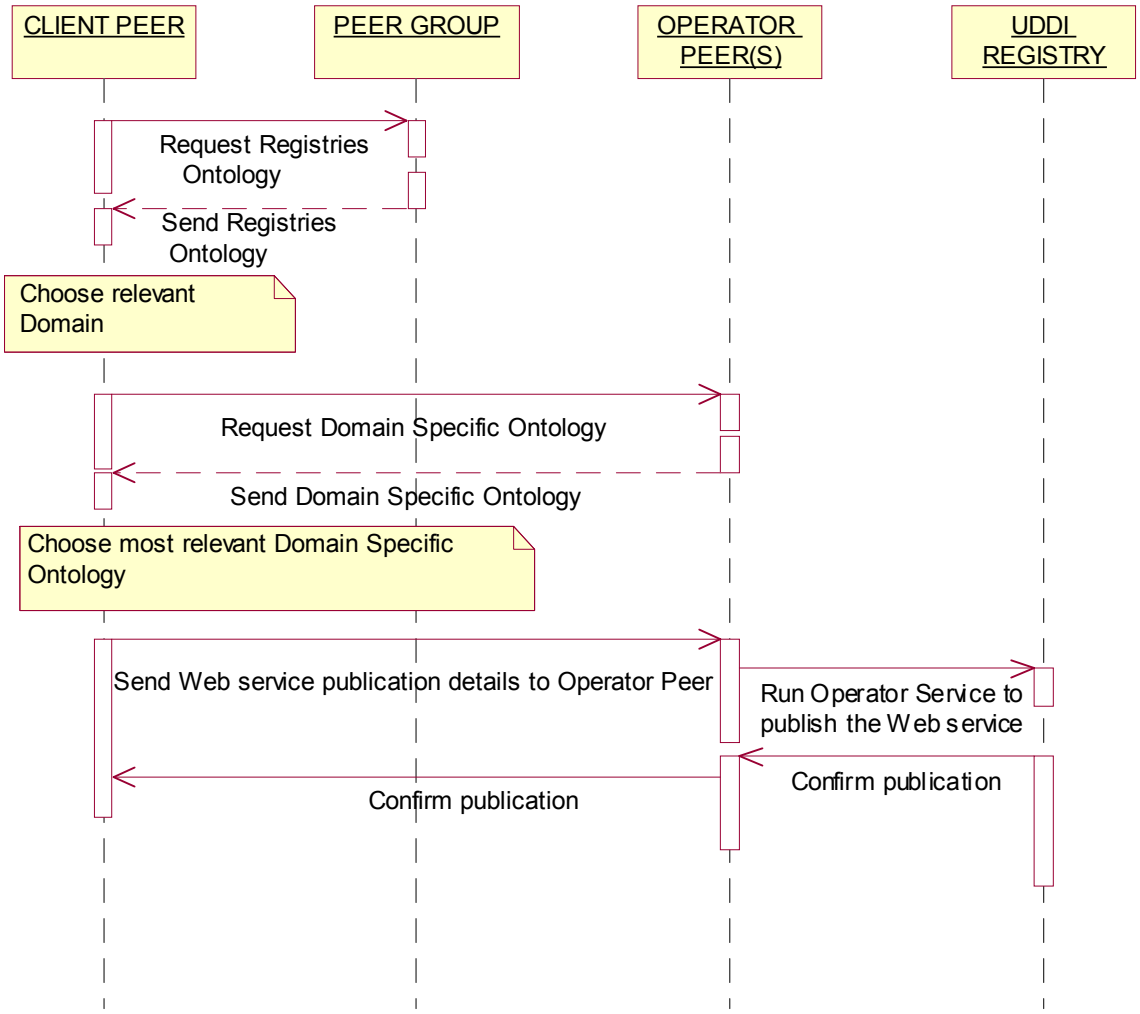


Figure 6: Interaction Diagram for Web Service Publication

4. Semantic Publication and Discovery

The key to enabling semantic discovery is adding semantic annotations to Web service specifications either in registries or service descriptions. Currently Web services are described using WSDL descriptions, which provide operational information. Although WSDL descriptions do not provide (or at least explicate) semantics, they do specifying the structure of message components using XML schema constructs. In this section we present two approaches for mapping these constructs to domain specific ontologies. Using these mappings, we intend to capture the meaning implied by the Web service provider in that domain. This additional information could be used to enable semantic discovery, if the user service requirements could also be expressed using concepts from the domain specific ontology.

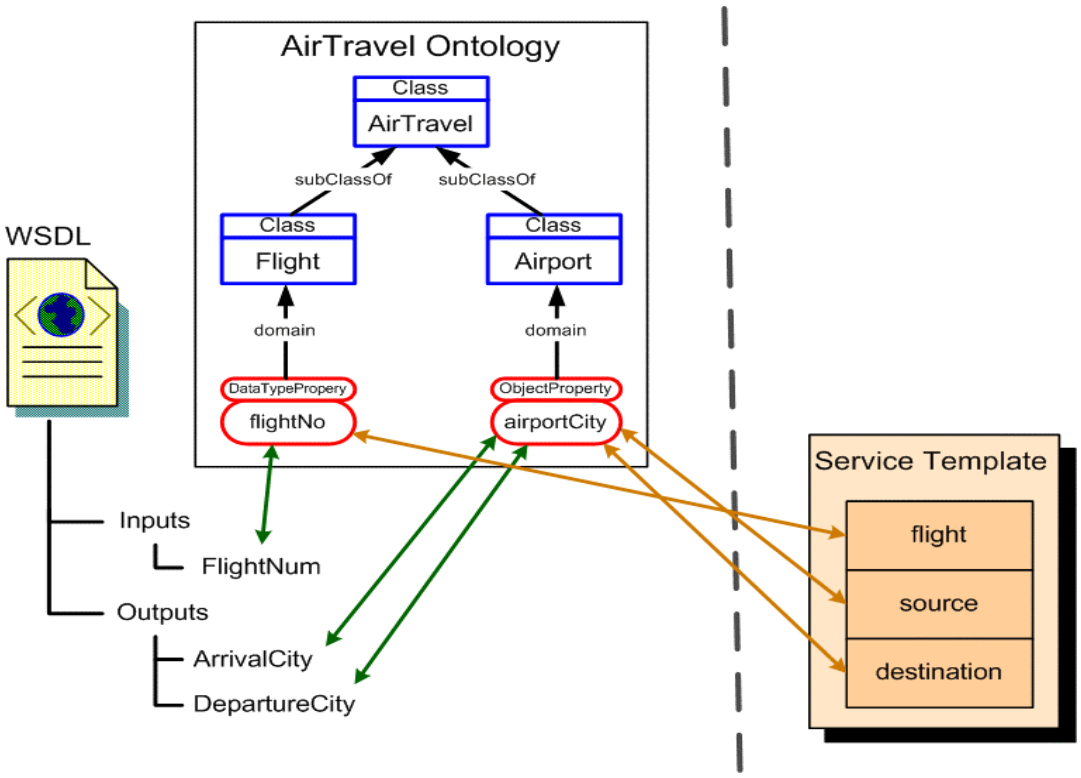


Figure 7: Semantic Publication and Discovery

Figure 7 shows the conceptual process of mapping WSDL concepts to the nodes in a domain specific ontology during service publication. It also depicts the creation of template using nodes in domain specific ontology for semantic discovery of services. As shown in the figure, the input concept of WSDL file *FlightNum* is mapped to the *flightNo* node in the *AirTravel ontology*. In addition, the output concepts *DepartureCity* and *ArrivalCity* are both mapped to the *airportCity* in the *AirTravel ontology*. These mappings can be used in the discovery process, by having the user map his discovery requirements to nodes in the domain specific ontology. This can be achieved by creating a template based on the concepts from the domain specific ontology.

The mappings between WSDL and the ontology are captured in UDDI using the *tModels and CategoryBags* [25]. *tModels* are metadata constructs in UDDI data structure that provide the ability to describe compliance with a specification, a concept or a shared understanding. They have various uses in UDDI registry. Commonly agreed specifications or taxonomies can be registered with UDDI as *tModels*. They can also be used to associate entities with individual nodes in taxonomies. When a *tModel* is registered with UDDI registry, it is assigned a unique key, which can be used by entities to refer to it. To categorize entities in UDDI, *tModels* are used in relation with *CategoryBags*, which are data structures that allow entities to be categorized according to one or more *tModels*.

To implement the semantic publication services using UDDI as an Operator service for a registry, two *tModels* have been created in that registry, one for representing the taxonomy of input concepts and the other for representing the taxonomy of output concepts. These *tModels*

are linked with the domain specific ontology using *overviewURL* tag of these *tModels*. During publication, the domain specific ontology concepts, along with the unique keys of the input and output *tModels* are used to semantically categorize the Web service. Using key-value pair property of *tModels*, these mappings can be stored in UDDI registries. The value would be the concept in the domain specific ontology and the name would be the key of the input or output *tModel*.

Two different types of mapping techniques used in Semantic publication and a discovery mechanism are explained in detail in the following sections.

4.1 Semantic Publication Service with Manual Mapping

The conceptual mapping discussed in section 4 can be achieved manually or using semi-automatic fashion. This section discusses semantic publication service that uses manual mapping. Figure 8 shows the interface used to manually map WSDL file concepts to the concepts of the domain specific ontology. In this GUI, the domain specific ontology is represented as taxonomy of concepts. The user can load the WSDL file of the service. The tool parses and displays it as a tree structure. The user can then manually map the input and outputs of the service to the nodes in the ontology. The service is then published in UDDI and it is semantically categorized using the mappings. These mappings are also stored in WSDL and we refer to semantically enhanced WSDL as annotated WSDL. For example, in Figure 8, the user maps the WSDL input concept *in0* of the message *getAirportInformationRequest* to the *AirportName* node in the ontology. During publication in UDDI, the Web service will be categorized with the input *tModel* and the concept *AirportName.* All search queries requesting inputs that have been mapped to *AirportName* will return this Web service. Typically, all inputs and outputs of a Web service should be matched to enable relevant searches for it.

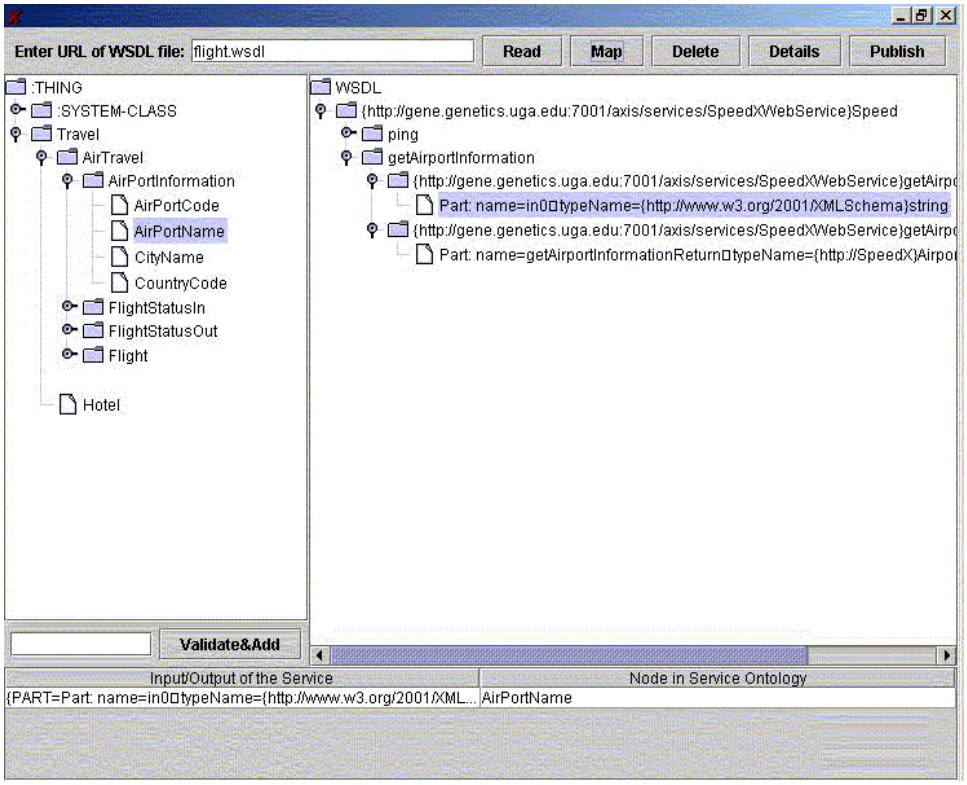


Figure 8: Screenshot of the Interface for Manual Mapping

4.2 Semantic Publication Service using Semi-automatic Mapping

This semantic publication service attempts to automate mapping between WSDL concepts and the domain specific ontologies. We have developed an algorithm SAWS [29] to automatically map each individual concept in the WSDL description to an ontological concept. Automation in this kind of mapping brings a number of difficulties. Primary reason for this difficulty is that XML schema does not support notion of classes and properties like ontologies. However, the structure of an XML element is hierarchical as elements in XML can have children. So, comparing with the ontological concept requires comparing not only element but also the hierarchical structure below it to the class and property structure of the ontological concepts.

The SAWS algorithm compares a concept from WSDL and an ontological concept and returns the degree of similarity (DS) between them. It is a combination of a structure matching algorithm and an element level matching algorithm. The element level matching algorithm calculates the linguistic similarity between the concepts whereas the structure matching algorithm considers the similarity between sub-tree of those concepts and calculates the structural similarity. The overall DS is then calculated as the geometric mean of the Structural similarity and Linguistic similarity of these two concepts. The degree of similarity is scaled on a scale of 0 to 1. Based on the degree of similarity, the user can accept or reject the mappings.

SAWS algorithm represents the schemas in the form of a graph which allows for a simple implementation of the structure matching algorithm based on DFS algorithm. The linguistic match algorithm is further divided into two steps namely preprocessing and concept matching. The preprocessing step implements techniques to remove suffixes to get morphological roots of the words, expand acronyms, tokenize words and thus create a set of parallel words using Wordnet. The second step calculates the actual match score. It tries to find if the words are synonyms, hypernyms or hyponyms with the set of parallel words acquired from preprocessing. In the case of absence of any parallel word it uses a substring matching algorithm based on the NGram matching algorithm.

Figure 10 shows a screenshot of the interface used for annotation. The interface provides the user with capabilities of specifying WSDL files and ontologies used for mapping. Subsequently our mapping algorithm is executed and recommended mappings are displayed to the user. The interface also provides the user the ability to accept, reject or modify these mappings. The user can also specify additional mappings. Finally, the mappings are written to the WSDL file as annotations. The modified WSDL file along with the original WSDL file is shown in figure 9.

```
<xsd:complexType name="Temperature">
- <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="ambient" type="xsd:double" />
    <xsd:element maxOccurs="1" minOccurs="1" name="dewpoint" type="xsd:double" />
    <xsd:element maxOccurs="1" minOccurs="1" name="relative_humidity" type="xsd:int" />
    <xsd:element maxOccurs="1" minOccurs="1" name="string" nillable="true" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Pressure">
- <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="altimeter" type="xsd:double" />
    <xsd:element maxOccurs="1" minOccurs="1" name="slp" type="xsd:double" />
    <xsd:element maxOccurs="1" minOccurs="1" name="delta" type="xsd:double" />
    <xsd:element maxOccurs="1" minOccurs="1" name="delta_hours" type="xsd:int" />
    <xsd:element maxOccurs="1" minOccurs="1" name="string" nillable="true" type="xsd:string" />
  </xsd:sequence>
</xsd:complexType>
```

Part of Original WSDL file

```
<xsd:element name="Temperature" DAML-Concept="weather-ont:DerivedTemperature"/>
<xsd:element name="Pressure"  DAML-Concept="weather-ont:PressureEvent"/>
```

Part of New Annotated WSDL file

Figure 9: Snippets of original and annotated WSDL files

The problem of mapping concepts in WSDL file to ontological concepts is similar to mapping two schemas. We realize that it is very difficult to map schemas automatically (e.g., see [26], [27]. One reason is that most schemas have some semantics which are not formally expressed and are only in the mind of designer.

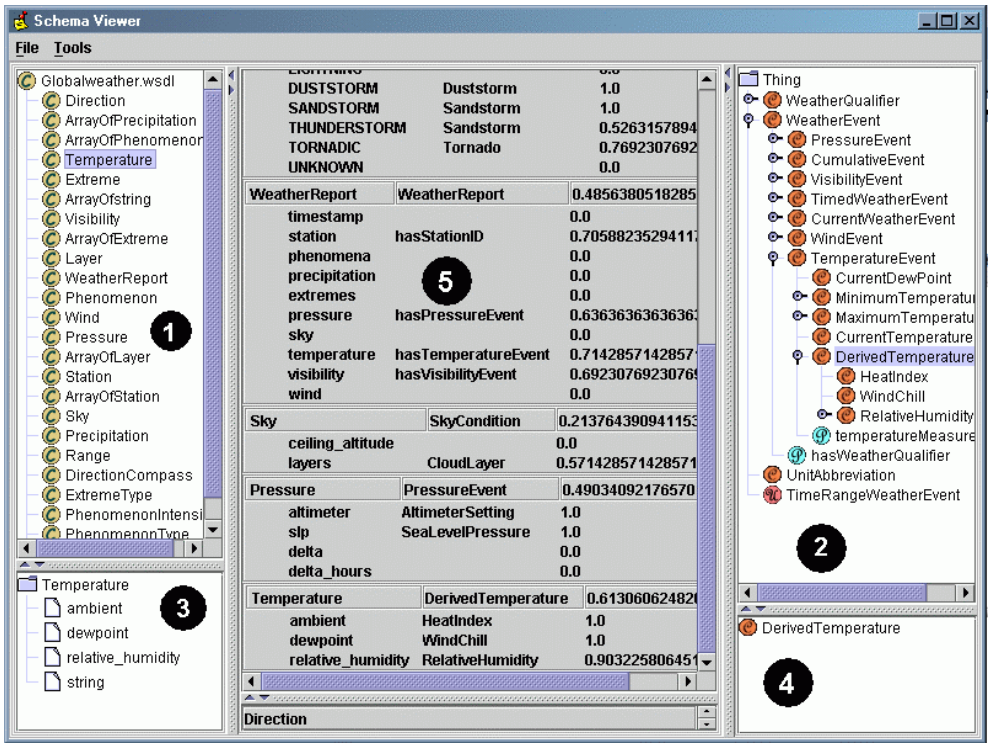


Figure 10: Screenshot of the Interface for Semi-automatic Mapping

In figure 10, the concept *temperature* in the WSDL schema is mapped to ontological concept *DerivedTemperature*. These concepts are shown in ❸ and ❹. The mapping shown in ❺ shows the mappings between components of these concepts e.g. *relative_humidity,* a component of the *temperature* concept, is matched with *RelativeHumidity*, a subclass of *DerivedTemperature*. ❶ and ❷ show all the concepts from WSDL description and ontology respectively. The figure also shows the degree of similarity measures between different concepts. After the semi-automatic mapping is completed, the service is published in UDDI and the mapping details are used to semantically categorize the service and

4.3 Semantic Discovery Service

Semantic discovery of services is done with the help of the mappings that were recorded in UDDI that is mentioned in sections 4.1 and 4.2. The semantic matching algorithm we have implemented is a simple algorithm to perform semantic discovery. The desired properties of the required Web service can be described using a service template (ST). A service template is created by specifying the inputs and outputs using concepts from the domain specific ontology. Matching of ST with the registered services is then carried out using the categorization details mentioned in 4.1. The results are sent back to the Client Peer.

5. Related work

ebXML version 3.0 which is not yet officially released by OASIS committee which is expected[6] to discusses the distributed registries model. This registry information model of the version [30] discusses supporting cooperating registries and registries federation. Co-operating registries imply that they are associated with each other, meaning that registry object reference can be across registries. "A registry federation is a group of registries that have voluntarily agreed to form a loosely coupled union. Such a federation may be based on common business interests and specialties that the registries may share. Registry federations appear as a single logical registry, to registry clients." The objective of this initiative shares some of the objectives of our work. MWSDI supports creating registry federation by grouping registries that are mapped to the same node in *Registries Ontology*. The registry federations discussed in [30] are based on the P2P model. In MWSDI too, the registries are considered as peers. The federated queries discussed in [30] can be executed in MWSDI using the registries ontology. Though this work seems to be closely associated with our work, it focuses mainly on the registry information model and discusses issues like object replication, object relocation and Lifecycle management for forming registry federation. In comparison, our work is not on the data structures of registries but focuses on building a scalable environment for publication and discovery across multiple registries. Our work also suggests protocols for Peer initiation and Client Interaction. In addition our work uses the *Registries Ontology* to maintain a global view of the registries, associated domains and uses this information during Web service publication and discovery.

Current research in Web services focuses on semantic Web services. Adding semantics to resources like Web services makes them machine processable [12]. The architecture that has been proposed in [31] discusses using semantics at different levels of Web services stack. They discuss having ontology servers and associated repositories to maintain domain concepts as

[6] The information is obtained from the ebxml-dev mailing lists available in http://www.ebxml.org

ontologies. In MWSDI, we provide similar functionalities using Operator Peers that maintain Web service registries and provide domain specific ontologies. Since domain specific ontologies provide a better conceptualization of a domain than general purpose ontologies the publication and discovery can be made more meaningful using them.

DAML-S ontology was created to enable the semantic description of Web services [32]. Recent work from DAML-S group [33] proposes using WSDL in addition to DAML-S description to completely describe a Web service In our work, we have annotated WSDL by associating its input and output types to domain specific ontologies As DAML-S is yet to get industry wide acceptance, we chose to use WSDL to add semantics to it. We have used UDDI structures to store the mappings of input and output types in WSDL files to domain specific ontologies. We have adopted this approach similar to the one suggested by [20]. This related work adds semantic matching capability to UDDI, by translating DAML-S representation of a service to UDDI representation so that it can be translated back to DAML-S representation for semantic matching of service specification. MWSDI provides similar functionality using WSDL, which we believe makes it more easily adoptable to the approach and standards the industry has chosen to accept. A detailed description of additional tags and annotations for adding semantics is provided in [25]. As the work to semi-automatically annotate a WSDL with *operation-ontology* mapping, preconditions and effects is underway, the publication of services discussed in section 4.1 and 4.2 and the discovery of services discussed in section 4.3 do not include *operation-ontology* mapping, preconditions and effects.

Using semantic metadata leads to the issues of scalable architectures for sharing, maintaining and distributing it. Peer-to-peer networks seem to provide an ideal environment for such systems. Peer-to-peer and Semantic Web issues are discussed in [34] [35]. [36] discusses using peer-to-peer, Semantic Web and Web services as enabling technologies to create a semantic driven service oriented architecture. Our work encompasses contributions from all these areas and provides peer-to-peer environment for Semantic Web service discovery and publication. Semantic gossiping [37] presents an architecture where mappings between schemas are used as a basis for query propagation. Its uses a bottom up approach for semantic agreement in a peer-to-peer environment, where there is no global ontology. However, while MWSDI allows registries to maintain their own domain specific ontologies or schemas, it uses a global ontology to maintain a relationship between registries. Since the premise of our work is to maintain relationships between registries, using *Registries Ontology* is critical. For semantic agreement, MWSDI allows registry providers to update the *Registries Ontology* to either relate their registries to existing concepts or to create their own concepts.

6. Conclusion and Future Work

We present techniques and prototype implementation of MWSDI. Our approach involves creating an infrastructure of registries for semantic publication and discovery of Web services. The primary motivation of our work is the expected growth in the number of registries and the lack of semantics in Web service representation. Our system provides a scalable architecture to access such registries. In addition, we provide semantic publication and discovery capabilities by using a domain specific ontology for each registry. We have presented two algorithms for

semantic publication and discovery using WSDL descriptions. Both these algorithms map inputs and outputs of Web services to ontological concepts. Subsequently, searching can be carried out using templates constructed using the ontological concepts.

In our approach, we treat a Web service as a black box having a set of inputs and a set of outputs. Annotating these inputs and outputs gives us a significant improvement in discovery and is better than the current approach used by UDDI. However each WSDL description may have a number of operations having different functionalities. Each operation would have its own set of inputs and outputs. For example, the same Web service may have operations for both selling and buying books. We believe our searching algorithms can be significantly improved by two techniques. Firstly the operations themselves should be mapped to concepts in the domain specific ontology which depict functionality. Secondly all inputs and outputs in the WSDL description should not only be mapped to concepts in the domain specific ontology but also grouped according to operations. The domain specific ontologies would have to be modified to maintain concepts that depict functionality along with the already existing input and output concepts. We are currently working on implementing this algorithm.

A significant part of this paper discusses implementation and architecture of the peer-to-peer network used by MWSDI. We discuss how using a peer-to-peer network gives us the scalability and flexibility required for creating an infrastructure for diverse Web service registries. We have tested our work with UDDI registry implementation provided in JWSDP [38]. However, this idea is applicable to any UDDI registry implementation and other type of Web services registries. Issues not covered in this paper that are planned as future enhancements are:

- Redistribution of service publication among registries
- Exchange of semantics between registries
- Full query support using all kinds of relationships among registries
- Adding reliability for the Gateway Peer by replication
- Automating registry selection in a domain using techniques for searching relevant ontologies discussed in [35]
- Study on performance and reliability of the P2P network and implementing security measures

According to UDDI, future specifications and features will aim to provide the ability to manage hierarchical business organizations, communities and trade groups. In addition, several Enterprises already have private registries and some companies have established an e-marketplace UDDI for the different domains. The infrastructure suggested in this paper can be used to support all these types of registries in a common environment for better service searching. MWSDI can also be adopted for enterprise level applications. For enterprises which have large number of departments, each having lots of Web services, the MWSDI can be chosen with each department running a department specific registry and each registry conforming to a department specific ontology or a common enterprise ontology.

From the business perspective MWSDI is all about grouping services and distributing them in different registries based on domain specialty, for locating the right services easily. On the other hand, from the technical perspective, MWSDI provides a scalable infrastructure for

accessing multiple registries and semantic enhancements to current service discovery mechanism. We believe that to develop processes in the current network economy [39], architectures like MWSDI will drive the evolution of businesses interactions using Web services. This infrastructure will also help Web services in changing the focus from static to more dynamic business settings.

References

[1] Universal Description, Discovery and Integration: UDDI Technical White paper. 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[2] E. Box et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000. Available at http://www.w3.org/TR/SOAP .

[3] E. Christensen et al., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, Available at http://www.w3.org/TR/wsdl

[4] http://ws.apache.org/axis/

[5] The Evolution of UDDI, UDDI.org White Paper, 2002 http://www.uddi.org/pubs/the_evolution_of_uddi_20020719.pdf

[6] L. Gallagher, L. Carnahan. A General Purpose Registry/Repository Information Model, Information Technology Laboratory National Institute of Standards and Technology. 2nd Draft - 23 October 2000.

[7] S. Macroibeaird et al. Using UDDI to find ebXML Regs/Reps, April 2001. http://lists.ebxml.org/archives/ebxml-regrep/200104/pdf00002.pdf

[8] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications." Knowledge Acquisition, 5(2), 199-220, 1993.

[9] F. Curbera, D. Ehnebuske, D. Rogers, Using WSDL in a UDDI Registry, Version 1.07, UDDI Best Practice, May 21, 2002. http://www.uddi.org/pubs/wsdlbestpractices-V1.07-Open-20020521.pdf

[10] R. Schollmeier, "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications." Proceedings of the First International Conference on Peer-to-Peer Computing, 2001.

[11] F. Curbera, W. Nagy and S. Weerawarana, "Web Services: Why and How." Workshop on Object-Oriented Web Services – OOPSLA 2001, Tampa, Florida, USA, October 2001.

[12] T. Berners-Lee, J. Hendler and O. Lassila. "The semantic web." Scientific American, 284(5):34–43, 2001.

[13] T. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, and E. Sandewall, editors, , San Mateo, CA, 1991. Morgan Kaufman

[14] V. Kashyap and A. Sheth. Semantics-based Information Brokering. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), November 1994.

[15] A. Sheth. Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman, editors, Interoperating Geographic Information Systems. Kluwer Publishers, 1998.

[16] A. Sheth, R. Meersman, Amicalola Report: Database and Information System Research Challenges and Opportunities in Semantic Web and Enterprises, SIGMOD Record 31 (4). December 2002.

[17] M. Dumas, J. O'Sullivan, M. Heravizadeh, D. Edmond, A. Hofstede, Towards a semantic framework for service description. In Proc. of the 9th Int. Conf. on Database Semantics, Hong-Kong, April 2001. Kluwer Academic Publishers.

[18] W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, M. Musen., Knowledge Modeling at the Millennium (The Design and Evolution of Protege-2000). 1999.

[19] Core Component Dictionary, ebXML Core Components, May 2001. http://www.ebxml.org/specs/ccDICT.pdf

[20] M. Paolucci et al., Importing the semantic web in UDDI. In Proceedings of E-Services and the Semantic Web Workshop, 2002.

[21] F. Gandon, Ontology Engineering: A Survey and a Return on Experience, Institut de Recherche en Informatique et Automatique (INRIA), March 2002, ISSN 0249-6399. ftp://ftp-sop.inria.fr/acacia/fgandon/FabienGandon_RR4396.pdf

[22] M. Denny, Ontology Building: A Survey of Editing Tools, November 06, 2002 http://www.xml.com/pub/a/2002/11/06/ontologies.html

[23] B. Benatallah, M. Dumas, M-C. Fauvet, F. Rabhi, Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi (Eds): "Patterns and Skeletons for Parallel and Distributed Computing". Springer Verlag (UK), 2002.

[24] L. Gong, "JXTA: A Network Programming Environment". IEEE Internet Computing, (5)3:88--95, May/June 2001.

[25] K. Sivashanmugam, K. Verma, A. Sheth, J. Miller, Adding Semantics to Web Services Standards, Proceedings of The International Conference on Web Services, pages 395-401, 2003.

[26] V. Kashyap, A. Sheth, Semantic and Schematic Similarities between Database Objects: A Context-based Approach, VLDB Journal, 5(4), 1996.

[27] J. Madhavan, P. A. Bernstein, and E. Rahm, Generic schema matching with cupid. In Proceedings of the 27th International Conferences on Very Large Databases, pages 49-58, 2001.

[28] J. Cardoso, A. Sheth, (2002) 'Semantic e-Workflow Composition', Technical Report, LSDIS Lab, Computer Science, University of Georgia, July 2002.

[29] A. Patil, S. Oundhakar, and A. Sheth, Semantic Annotation of Web Services, Technical Report, LSDIS Lab, Department of Computer Science, University of Georgia, March 2003.

[30] OASIS/ebXML Registry Services Specification v2.35: Committee Working Draft, OASIS, ebXML Registry Technical Committee, Feb 2003.
http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/ebxmlrr/ebxmlrr-spec/doc/ebRS.doc

[31] C. Bussler, D. Fensel, A. Maedche, A Conceptual Architecture for Semantic Web Enabled Web Services, SIGMOD Record 31(4), pages 24-29, 2002.

[32] A. Ankolekar et al., 'DAML-S: Web Service Description for the Semantic Web', in International Semantic Web Conference, Sardinia, Italy, pp. 348-363, 2002.

[33] DAML-S: Semantic Markup for Web Services, The DAML Services Coalition, http://www.daml.org/services/daml-s/0.7/daml-s.pdf

[34] M. Schlosser et al, A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services, (2002).

[35] M. Arumugam, A. Sheth, and I. B. Arpinar, Towards Peer-to-Peer Semantic Web: A Distributed Environment for Sharing Semantic Knowledge on the Web (2001).

[36] A.Maedche and S.Staab, Services on the Move - Towards P2P-Enabled Semantic Web Services, Proceedings of the Tenth International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki 2003/01/31.

[37] K. Aberer, P. Cudré-Mauroux, M. Hauswirth A Framework for Semantic Gossiping, SIGMOD Record, 31(4), 2002.

[38] Java Web Services Developer Pack, http://java.sun.com/webservices/webservicespack.html

[39] A. Sheth, W. Aalst, I. B. Arpinar, Processes Driving the Networked Economy, IEEE Concurrency, pp. 18-31, July-September 1999 (Vol. 7, No. 3).