# Towards A Framework for Service-Oriented Architecture Metadata Management

Malik M. Umar
Senior Software Engineer
SecureKey Technologies Inc.
Toronto, Ontario M2P 2E9, Canada
e-mail: UmarMalik@hotmail.com

Mohammad Alshayeb
Information and Computer Science Department
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
e-mail: alshayeb@kfupm.edu.sa

*Abstract*—**Service-Oriented Architecture has gained considerable attention in construction of enterprise level business solutions. Although this architectural approach provides many benefits, it comes at the cost of increased complexity. This research focuses on the development of a metadata framework using semantic web technologies of XML, Resource Description Framework (RDF) and Web Ontology Language (OWL). The resultant framework is in the form of an Enterprise Services Ontology and Enterprise Services Profile. These ontologies provide the foundation that has been employed to develop a proof-of-concept Services Repository to prove that a semantic service repository can be a viable means of addressing complexity and management problems faced by enterprise scale Service Oriented Architecture (SOA) implementations.**

*Index Terms*—**Service-Oriented Architecture, Metadata Management, Enterprise Services Ontology and Enterprise Services, acceptance criteria**

## I. INTRODUCTION

To varying degrees, most enterprises are using IT to improve business process, boost productivity, and increase customer satisfaction, all while holding down costs. At the same time, conventional wisdom holds that enterprise software strategies are no longer about installing new application silos [1]. Today, the application software business is about leveraging existing applications, data assets, and services, by integrating them to create a more seamless whole that serves the business [2]. Simply put, application integration, interoperability, and security are the most significant obstacles to the long-term vision of a business-driven, utility-oriented computing environment.

Just as the enterprise itself is becoming more virtual, the application platform is becoming more virtual as well [3, 4]. In the past, application silos met business needs because the application context they addressed was relatively self-contained. Today, however, organizations are going virtual in multiple dimensions. For critical systems that support business process, the application context is no longer a self-contained universe [1]. The typical business application context is growing to serve large numbers of customers, to integrate partners and suppliers and to enable an increasingly mobile workforce.

One of the most significant developments in moving to virtual enterprise has been the advent of Service-Oriented Architecture [5, 6] based on Web Services [1, 7-9]. Although the idea of Service-Oriented Architecture (SOA) has been around for some time, it has truly reached its potential for realization with the advent of Services based on XML over HTTP, commonly known as Web Services. Web Services is an industry accepted standard for implementing distributed components as services [5, 9]. Use of Web Services to implement a services based architecture has the unique advantage of open interoperability, which is highly desirable trait for a flexible architecture. SOA is a refinement of ideas previously presented in the distributed computing and modular programming. A Service-Oriented Architecture facilitates construction of applications using existing services. The main objective of this form of application architecture is to assist in creation of ad-hoc applications, developed primarily using existing functionality. The ultimate promise of SOA is that cost of application development is steadily reduced as most of the software required already exists and only orchestration is required to produce a new application.

As with any new technology there is a refinement process required to attain its full potential. SOA based on web services brings with it a new set of challenges that need to be addressed. This work is dedicated to improving the controls and management of Enterprise scale SOA, by controlling Enterprise Service metadata.

Service-Oriented Architecture based on Web Services has evolved to become architecture of choice for organizations to deal with challenges of cost, time, quality and interoperability. Although the properties of Web Services are ideally suited for a Service-Oriented Architecture and promote rapid application development, they also have certain overheads and drawbacks that need to be considered. Some of the limitations and shortcomings of a Web Service based SOA include:

- Systems developed via Web Services require consumer and provider to agree on semantics in advance [10].
- Web services lack an established mechanism for distributing service implementation information.
- Applications built on multiple layers of services represent a challenge for Enterprise Architects, to identifying critical junctions and dependencies.

These shortcomings of Web Services based Architectures represent unique challenges for business analysts, architects and enterprise application designers.

With the advent of web services as a viable platform independent technology, realization of a service-oriented architecture embodying the properties of platform independence, dynamic invocation and self-containment has become possible. Services available for attaining various types of functionality can be discovered by means of Universal Description, Discovery and Integration (UDDI) registries [11] and applications can quickly be built by utilizing pre-built services. New services can even be dynamically composed [12] based on user request. UDDI provides a simple mechanism for service consumer to look-up services from the registry, based on key word or category.

This level of indirection between the service consumer and provider allows for flexibility, but at the same time increases application architectural complexity. Because of this middle level of indirection, there is no mechanism available to identify all the consumers of a service. This might not affect applications that are acting as information aggregators but would have significant impact if the application relies on coarse-grained business services, which in turn are dependent on other services. Dependency information becomes especially critical while architecting new solutions or maintaining existing services infrastructure. This is especially true for large enterprise architectures that are composed of a heterogeneous collection of systems, interacting to provide business functionality.

Complexity cannot be eliminated in a dynamic evolving environment so the focus has to be diverted to managing this change and complexity, to balance the need of business for dynamism and the Enterprise Architect's need for stability. Mechanisms have to be developed to retain information about the service-oriented architecture, the relationship among services, their reliability, changeability and dependencies. These and other pieces (service configuration) of SOA metadata would give the Architects the following benefits:

- High-level service oriented view of the enterprise software systems
- Identification of interdependencies within services to discover critical services
- Assess risk and impact of change in the service ecosystem
- Centralized repository for all scattered service related information
- Support in managing daily operations and planning upgrades

Complexity and lack of transparency are major problems for enterprise architectures. A detailed literature review of current registry and repositories such as Description, Discovery and Integration (UDDI) &ebXML[13, 14], reveals that these technologies address the problems only partially. This work is aimed at alleviating the shortcomings of a Web Services based SOA, by harnessing advantages offered by semantic web technologies.

The main objective of this paper is to devise a framework that caters for persisting important dynamic and static metadata related to Web Services, thereby addressing the problems discussed above. Furthermore, we intend to develop a Web Services ontology containing the most pertinent information, based on our literature review. Finally using the theoretical basis of our web services metadata management framework, devise a registry-repository hybrid, as a proof of concept.

## II. Literature Review

Service-Oriented Architecture is a vast area of research and development to address some of its shortcomings and emerging requirements. Major work being done on SOA can be broadly classified into two categories: SOA Governance and SOA Infrastructure.

### *A. SOA Governance*

In order for enterprises to effectively adopt SOA and prevent the problems that plagued its predecessors from resurfacing, it is incumbent on SOA practitioner to use governance strategies different from traditional IT governance model. SOA governance is a key to the successful adoption of SOA. A governance model built and operated to govern the life cycle of services is essential for an SOA to deliver the benefits that make SOA attractive. Next, we examine the most important works in this area that are enhancing SOA governance practices and augmenting its implementations.

#### *A.1. Policy Management*

MacKenzie et al. [15] defined service policy as a set of assertions that express intent on the part of a participant, which could be applied to many aspects of a Service-Oriented Architecture, such as quality of service, manageability security etc. The correctness service policy ensures the correctness of system behavior. The enforcement of service policy thus far is limited to the runtime behavior of services.

To provide policy enforcement during design time, Zhou et al. [16] proposed the use of logical policy model expressed as UML as input and to produce a physical service policy model mapped to physical system topology. Furthermore, this process would also generate service deployment model that describes the relation among policy artifacts, and guides and automates the deployment of policies in runtime environment.

### *A.2. Service Development Process*

The Service Development Process in SOA also falls under the umbrella of Governance. Services developed as a part of an Enterprise SOA should adhere to best practices and principles of this architectural style and development governance strategies ensure this. Lianjun[17] proposed a Business Services Modeling & Analysis (BSMA) technique to help architects in defining the elements in each of the SOA layers and making critical architectural decisions at each level. The BSMA modeling process consists of the following stages: identifying business goals & artifacts, identifying business services, allocating business service functionality and component interaction & information analysis.

Information gathered during these phases is aggregated by System Dynamics tool to generate various impact scenarios of introducing the service and its interaction with the service ecosystem.

### *B. SOA Infrastructure & Web Services*

Web Service registry and repositories are an integral part of any enterprise class SOA implementation [5-7, 18, 19] and there has been considerable effort in enhancing their capabilities to address problems of management and complexity. There are two main standards for Web Service registry-repository; namely UDDI &ebXML. UDDI is considered the industry standard and has the most wide spread adoption. ebXML is gaining momentum but still lacks major vendor adoption and support.

### *C. Web Service Registry & Repositories*

Current UDDI specifications mainly focus on service registry and discovery by service consumers. There is room for improvements, to address some of the problems of service metadata management. Following is a review of significant research efforts in this area.

### *C.1. Service Discovery*

Discovery of appropriate services by interested consumers is one of the most challenging areas in the wide scale adoption of Web Services. UDDI provides a limited set of attributes that can be interrogated to select a service. These include service name (given by service provider), key reference (unique id) and category bag (business categories to which the service belongs). These do not provide sufficient information for service selection and often requires a Service to be selected at design time rather than dynamically being selected at runtime.

To overcome these deficiencies with service discovery ShaikhAli et al. [20] prescribed modifications to UDDI, by extending bussinessClass to include additional arbitrary properties. They further propose to enhance the UDDI querying mechanism to cater for conditional logic. These changes extended the UDDI while preserving backward compatibility, at the same time providing better matching capabilities. .

The service discovery problem is tackled by Liu et al. [21] by means of a graph search based on service discovery algorithm [22]. This approach also involves adding auxiliary data structures, shown inTable Iand UDDI APIs to the traditional UDDI without affecting the original capabilities.

TABLE I.
ATTRIBUTES TO SUPPORT SERVICE DISCOVERY IN A MODIFIED UDDI [21]

| Field name | Field type | Field length | Is primary key? | Is Null? |
|---|---|---|---|---|
| BUSSINESS_KEY | VARCHAR | 255 | Yes | No |
| SERVICE_KEY | VARCHAR | 255 | Yes | No |
| LINK | TEXT | | No | Yes |
| NAME | VARCHAR | 255 | No | No |

Lee et al. [23] used a slightly different approach, by proposing a framework to include the Web Services Description Language (WSDL) file in the service search query and give a match ranking, along with possible alternatives. This adds another dimension to service search query including within it the communication interface exposed by the service. With the above mentioned approaches service discovery improvements are attained, but service discovery based on semantics is still lacking. Furthermore, these techniques do not naturally lend to automated discovery and selection

### *C.2. Quality of Service (QoS)*

Quality of Service in the context of Web Services refers to non-functional properties of Web Services such as performance, reliability, availability and security [24]. These attributes play an important role in the selection of Web Services and may influence selection of one service over another.

The UDDI model does not provide any facility to accommodate QoS attributes. Similar to data structure changes proposed by ShaikhAli et al. [20] and Liu et al. [21] Liu [24] proposed linking a Service Attribute Schema with UDDI registry to retain QoS data.

Liu [25] proposed another important change that is the modification of the service registration process with the UDDI to incorporate solicitation of QoS data by the service provider.

### *C.3. Web Service Semantics*

UDDI or ebXML provide an efficient mechanism for web services to be registered for discovery and WSDL publishes the communication contract of services. What is lacking in this setup is the nonfunctional data pertaining to the service and the semantics behind it.

Dogac et al. [26] proposed modification to ebXML Registry Information Model RIM [13] to support semantics by storing OWL ontology in the ebXML registry, while at the same time persevering backward compatibility. The technique proposed used ebXML Classification Hierarchies [13] to represent OWL Ontologies. Although Dogac achieved the stated goal of adding semantics to ebXML registries, but this was accomplished at the cost of design ambiguity and use of ebXML constructs in a manner they were not intended to be used by authors of the standard.

Roh et al. [27] described modifications to the architecture of the ebXML registry to accommodate OWL ontologies. Roh proposes a new information model called the semantic information model (SIM) for the sole purpose of storing semantic data, leaving the ebXML registry model unchanged. SIM consists of classes designed specifically to model OWL classes & their attributes. The limiting factor in this approach is clients would require being aware of ebXML search and retrieval mechanism to get the semantic data.

UDDI provides the tModel framework for extensibility and flexibility. tModel are pointers to external resources and consist of a tModelKey (unique identifier) keyValue and keyName (descriptive name). Luo et al. [28] proposed the use of the tModels to represent all constructs of OWL. The translation from OWL ontology to a UDDI tModel hierarchy is achieved through an external client side component, eliminating the need to modify UDDI. Feng et al. [29] provided an extension to UDDI on service semantics facet.

Baker et al. [30] proposed a new approach within existing SOA methodologies that supports source-codesemantic flexibility via an intermediary Meta Data Layer (MDL), providing a layer of separation between the source code and the service. Virgilio[31] proposed a description of WSDL documents into a metamodel representation of Semantic Models that allows interoperability at different levels of abstraction.

From research into modification of registries to support semantic capabilities that the possible approaches to solve this problem fall under two categories:

- Use or modification of registry/repository internal annotation framework to accommodate ontologies [26-28].
- Storing ontology information in external databases and redirecting queries to external matching modules [32].

### *C.4. Web Ontology Language – Services (OWL-S)*

OWL-S [33] has gained the status of a standard for defining web service semantics. It consists of a core set of markup language constructs for describing the properties and capabilities of Web Services in a computer interpretable form. It is essentially an ontology for services built on top of Web Ontology Language (OWL) [34]. OWL-S is designed to facilitate the discovery, invocation, composition and interoperation of web services. We have already reviewed the major works to facilitate service discovery [26-28, 32]. Here we review the important developments related with service invocation, composition and interoperability with OWL.

To overcome the deficiencies in the semantic modeling of dynamic service composition with OWL-S, Li et al. [35] researched the use of AI planning and Description Logic DL. The proposed solution extends the OWL-S model by providing a service composition mechanism that takes into account user preferences. This solution relies on representation and reasoning capabilities of DL along with modeling faculties of action state transformation of AI planning.

OWL-S facilitates the invocation of services by means of Service Groundings [34]. A service can support multiple grounding meaning it can support various protocols and mechanism for its invocation simultaneously. To further enhance this facility of OWL-S Gannod et al. [36] proposed a general framework that takes inputs during the service design (MDA artifacts) to construct the OWL-S profile and grounding. Bingxian and Xie[37] proposed a method to describe process model of OWL-S based semantic web service with PNML and OWL.

The objective of this framework is to use design artifacts to generate OWL-S grounding and in the second phase use service groundings as the contract against with service realization occurs.

## III. Enterprise Services Metadata Management Framework

This section provides a foundation framework for developing an Enterprise Services Metadata repository using Semantic Web technologies. In developing this framework, we made use of Web Ontology Language (OWL) and resource description framework (RDF). Using semantic Web markup allows us to persist enterprise services metadata in a format that is readable by both machines and humans. Furthermore, this approach also facilitates use of existing well-established ontologies in developing a rich framework for enterprise services metadata management.

To create the enterprise services metadata management framework we developed an ontology that is an extension of OWL-S (Web ontology language for services). OWL-S allows us to make the most of existing capabilities of this ontology whilst at the same time having metadata that is required by enterprise repositories. Figure 1shows the hierarchy of various semantic markup languages and position of this work in the scheme or markup languages.

The following section describes how the OWL-S ontology has been modified to accommodate our objectives. OWL-S is based on three sections, which are:

1. Service profile
2. Service grounding
3. Service model

These three sub-ontologies can be used to describe any software services not necessarily web services. The service profile represents what service does, service model describes how a service works and service grounding deals with how to access the service. The following section presents the modification of the service profile to incorporate metadata for enterprise services.

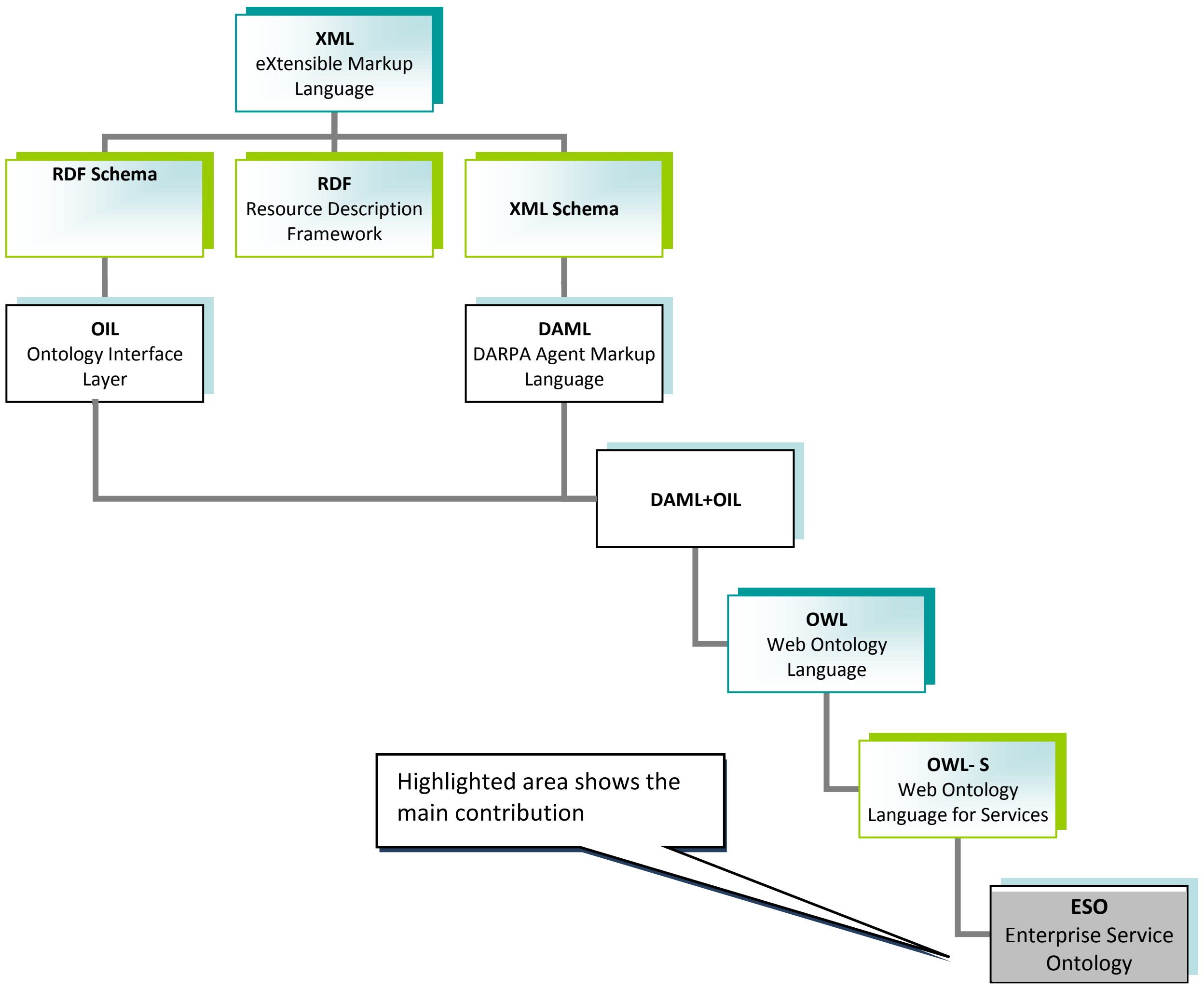

XML
eXtensible Markup Language
RDF Schema
RDF
Resource Description Framework
XML Schema
OIL
Ontology Interface Layer
DAML
DARPA Agent Markup Language
DAML+OIL
OWL
Web Ontology Language
OWL- S
Web Ontology Language for Services
Highlighted area shows the main contribution
ESO
Enterprise Service Ontology

Figure 1.Semantic Markup Language Hierarchy

## A. *Enterprise Service Profile*

This section presents a detailed description for an Enterprise Service Profile, based on the OWL-S Profile (as shown in Figure 1), incorporating significant metadata elements for managing an Enterprise Services Architecture shown in Figure 2 and Figure 3. In the following sections, all the elements of this new profile are discussed along with their RDF/OWL representation in graphical format.

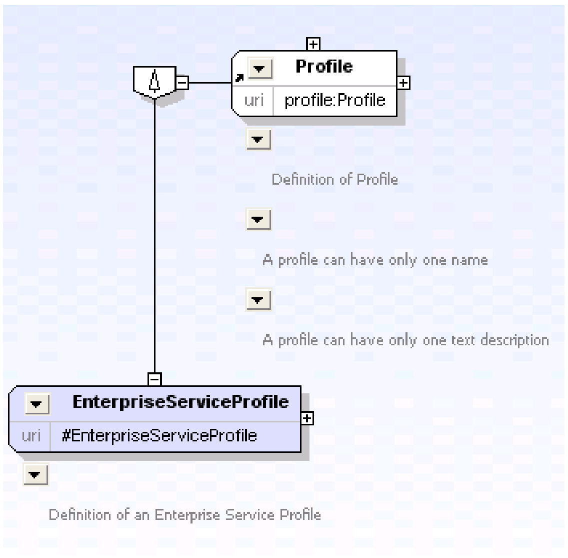

Profile
uri profile:Profile
Definition of Profile
A profile can have only one name
A profile can have only one text description
EnterpriseServiceProfile
uri #EnterpriseServiceProfile
Definition of an Enterprise Service Profile

Figure 2. Enterprise Service Profile Ontology extending from Profile Ontology

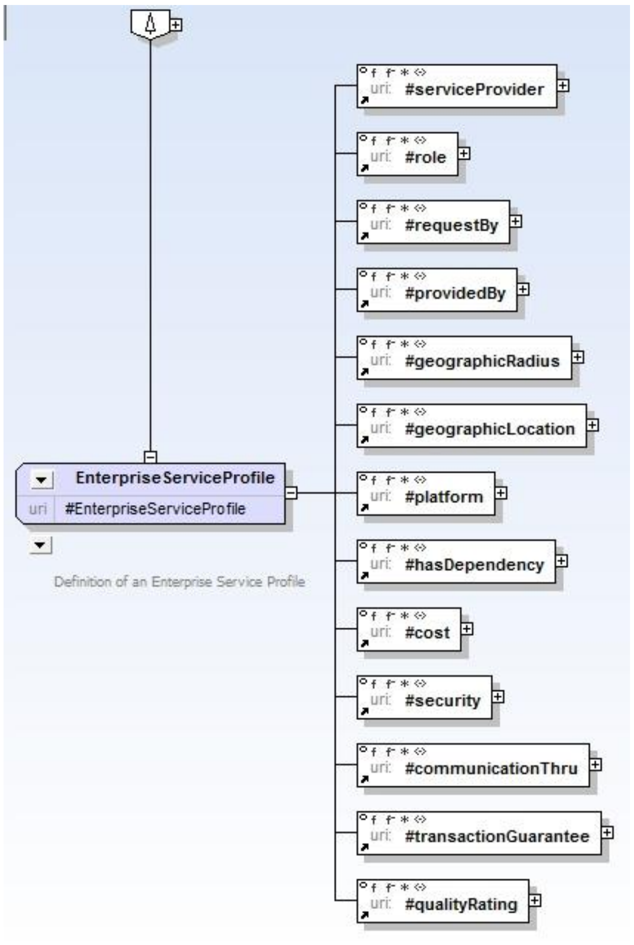


Figure 3. Enterprise Service Profile with associated metadata attributes

**Attribute Name:** Service provider (actor)
**Definition:** Service Hosting Entity
**Description:** Service Provider is the hosting entity where the service resides. In the context of an Organization, this unit is providing a runtime environment for the service. The provider in not necessarily the owner of the service and to successfully invoke the service at runtime the permission of the Owner may be required. For more detail on Owner & Permissions see the relevant sections in this section (Requested By, Role & Security).
Following is a list of properties for Service Provider. See Figure 4 for a graphical representation.
1. Name
2.Type
3.Phone
4.Fax
5.Email
6.Web Address
7.Physical Address

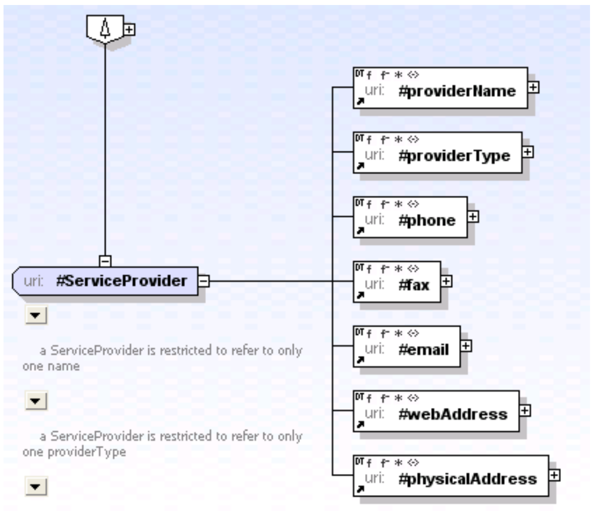


Figure 4. Enterprise Service Profile attributes - Service Provider

**Attribute Name:** Role
**Definition:** Business role for accessing service
**Description:** To invoke ES services, the service client must be within a certain role as mandated by the service developer. This role can have an associated realm for with the role is valid, such as Active Directory, LDAP or some Enterprise Identity Management System as shown in Figure 5.
**Rationale:** This attribute is required for security and enforcing business rules.

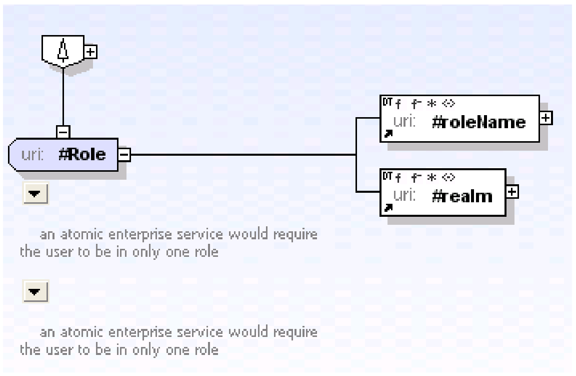

uri: #Role
an atomic enterprise service would require the user to be in only one role
an atomic enterprise service would require the user to be in only one role
uri: #roleName
uri: #realm

Figure 5. Enterprise Service Profile attributes – Role

**Attribute Name:** Requested By
**Definition:** Original or primary service customer
**Description:** Enterprise Services are created based on the request of a proponent - usually the primary user of the service. This is an important criterion in maintaining enterprise services architecture and to identify primary user, hence it has been included in the Enterprise Service Profile. To represent this we will make use of the Default Actor ontology (Part of OWL-S). Complete attributes of this class can be seen in Figure 6.

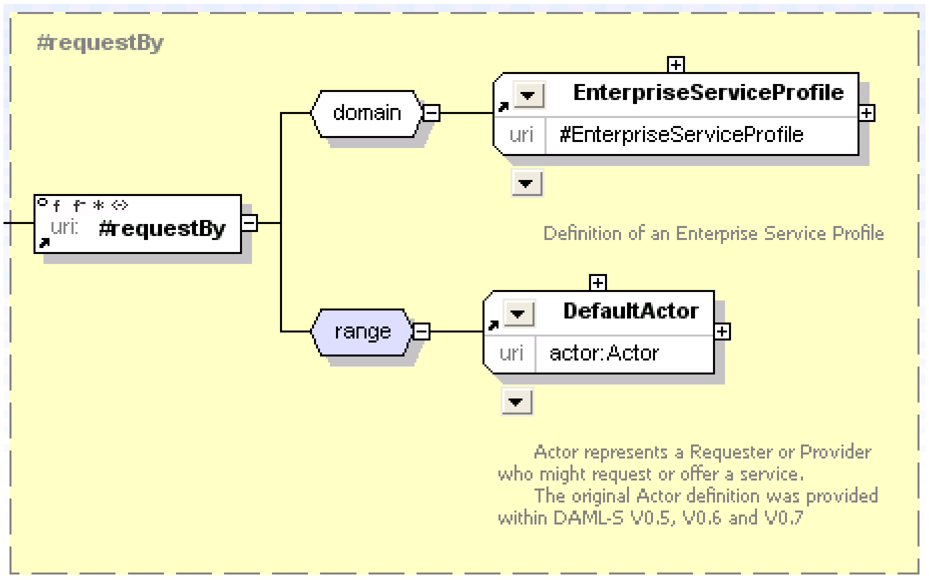

#requestBy
uri: #requestBy
domain
range
EnterpriseServiceProfile
uri #EnterpriseServiceProfile
Definition of an Enterprise Service Profile
DefaultActor
uri actor:Actor
Actor represents a Requester or Provider who might request or offer a service.
The original Actor definition was provided within DAML-S V0.5, V0.6 and V0.7

Figure 6.Enterprise Service Profile attributes - Requested By

**Attribute Name:** Provided By
**Definition:** Service Developer
**Description:** Although the requester and provider share the same range (as shown in Figure 6), namely the Actor Class form ActorDefault ontology. Having booth these types as part of the Enterprise Service Profile mandates that information about these elements be provided. This is necessary to identify primary consumer and developer of the service as this is an important criteria in service selection for prospective service clients.
**Rationale:** Inclusion is necessary to identify the builder or a particular service. In enterprise environments, service provider (host), request and developer are usually distinct entities.

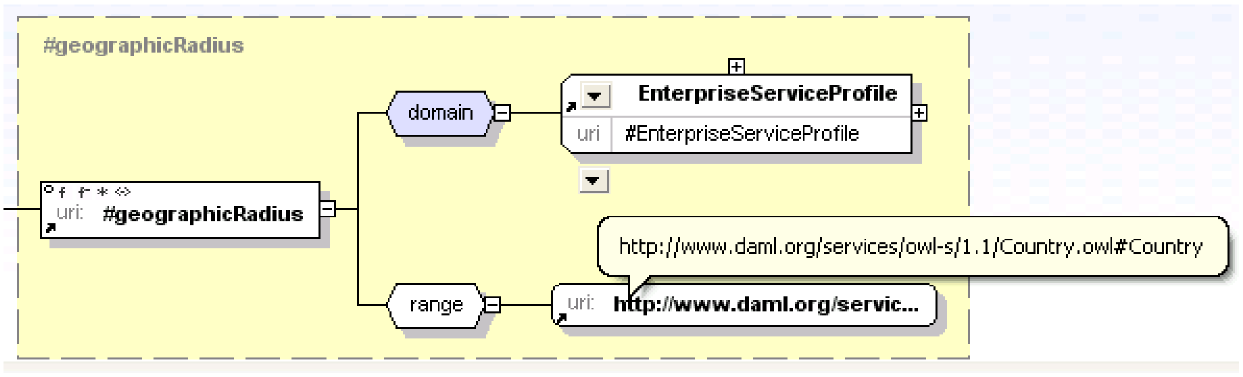

#geographicRadius
uri: #geographicRadius
domain
range
EnterpriseServiceProfile
uri #EnterpriseServiceProfile
http://www.daml.org/services/owl-s/1.1/Country.owl#Country
uri: http://www.daml.org/servic...

Figure 7. Enterprise Service Profile attributes - Geographic Radius

**Attribute Name:** Geographic Radius
**Definition:** Country or group of countries for which the service is applicable
**Description:** To define the geographic radius of a service to which it is applicable. This attribute has been defined as optional because a service provider may not restrict the use of a service within an area. Figure 7 shows that the range for this attribute has been defined using the Country ontology
**Rationale:** Some services are only valid or legally usable within a given geographic region. Inclusion of this attribute informs the service consumer of this limitation.

**Attribute Name:** Geographic Location
**Definition:** Location where the service resides
**Description:** Similar to geographic radius, location represents the location(s) where the service is hosted.
**Rationale:** Services that are located in very distant locations incur a network communication penalty. Inclusion of this attribute will allow consumers to select the most appropriate service based on network delay tolerance.

**Attribute Name:** Platform

**Definition:** Service hosting environment
**Description:** Although the platform that a service is usually not of interest to clients but that is not the case with Enterprise Services clients. A service may be selected for use within the enterprise based on the type & version of its host platform.
Properties of this class include:
1. Platform Name (J2EE, SAP, .NET etc.)
2. Version

See Figure 8 given below for graphical representation of this class.

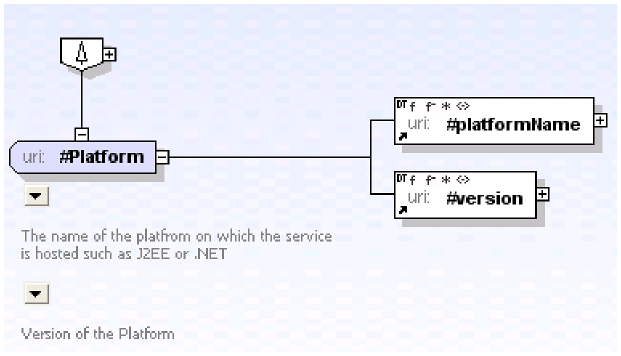

uri: #Platform
uri: #platformName
uri: #version
The name of the platfrom on which the service is hosted such as J2EE or .NET
Version of the Platform

Figure 8.Enterprise Service Profile attributes – Platform

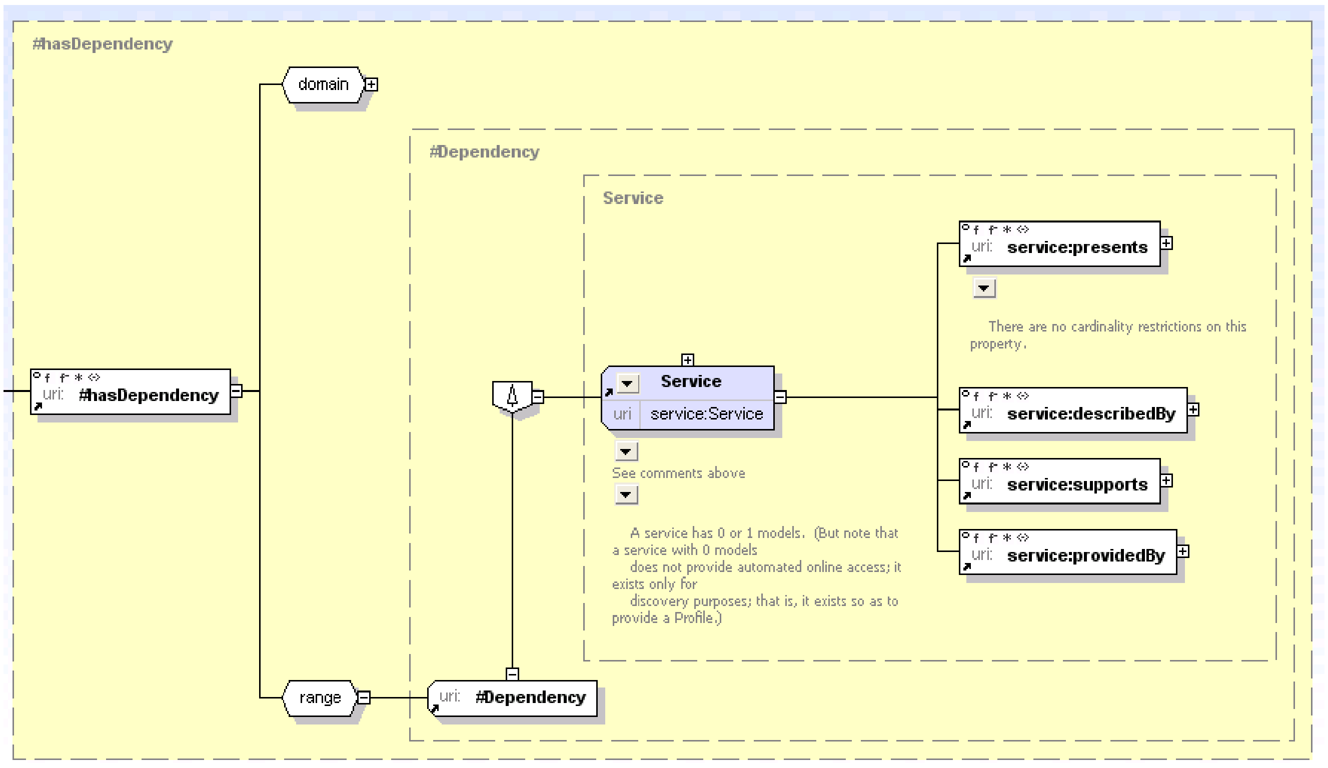

#hasDependency
domain
#Dependency
Service
uri: service:presents
There are no cardinality restrictions on this property.
uri: #hasDependency
Service
uri service:Service
uri: service:describedBy
See comments above
uri: service:supports
A service has 0 or 1 models. (But note that a service with 0 models does not provide automated online access; it exists only for discovery purposes; that is, it exists so as to provide a Profile.)
uri: service:providedBy
range
uri: #Dependency

Figure 9. Enterprise Service Profile attributes - Has Dependency

**Attribute Name:** Dependencies (#hasDependency)
**Definition:** Other services which are required for this service to execute.
**Description:** Enterprise web services can be compound in nature and depend on other services to complete their functionality. Dependency information provides a clearer picture of the overall implementation and relationship between services. Dependencies can be defined by referring to the Service ontology of the underlying services Figure 9 show the complete relationship between this attribute and ontology (Service) from which it is derived.
**Rationale:** Enterprise architects and other stakeholders responsible for managing the SOA infrastructure require this information to perform development and maintenance tasks.

**Attribute Name:** Cost
**Definition:** Cost of invoking service
**Description:** This is the cost of invoking the service. Service providers can charge for invoking their services. Cost has been introduced as a simple type here, but it is not necessary the cost is only in monetary terms. For example internal service may not incur a cost in dollars to execute, but may have other associated costs such as grid resource utilization, bandwidth consumption, persistence usage etc.

**Attribute Name:** Security
**Definition:** Security requirements for accessing service
**Description:** Security is a critical aspect of any Enterprise Service and there are many characteristics of security implementation that a client might be interested in. Such as protocol used for security, credentials, security algorithm, security assurance etc. To augment the Enterprise Service Profile Ontology we make use of the NRL ontology. The NRL ontology provides the

necessary basis for attaching the necessary metadata and draws on an existing well-established ontology.
There are two main aspects of security namely:
1. Security Objective
2. Security Concept

These two categories provide a basis for organizing all characteristic of web services security.
Security Objective has the following sub classes
1. Confidentiality
2. Availability
3. User Authentication
4. Message Authentication
5. Authorization
6. Message Integrity
7. Key Management
8. Replay Prevention
9. Trust
10. Host Trust
11. Covert Channel Prevention
12. Separation
13. Traffic Hiding
14. Anonymity

The Security Ontology defines following Security Concepts that support one or more security objectives defined above.
1. Security Mechanism
    a. Service Mechanism
    b. Host Mechanism
    c. Network Mechanism
    d. Application Mechanism
2. Security Policy
    a. Commercial Policy
    b. Military Policy
3. Security Protocol
    a. Signature Protocol
    b. Authentication Protocol
    c. Net Security Protocol
    d. Encryption Protocol
    e. Key Management Protocol

**Attribute Name:** Communication Through
**Definition:** Intermediary or gateway for service access
**Description:** Enterprise Web service interaction with the client can be either direct or in some instances the services interact with clients via intermediaries, such as Enterprise Service Bus. This requirement is usually due to security and non-repudiation purposes. As Figure 10 shows the range for this attribute is defined as OWL Thing, therefore, it is not restricted and the user of this ontology can define required values for this attribute.
**Rationale:** Included in Enterprise Service Profile to give complete path access path information from consumer to service.

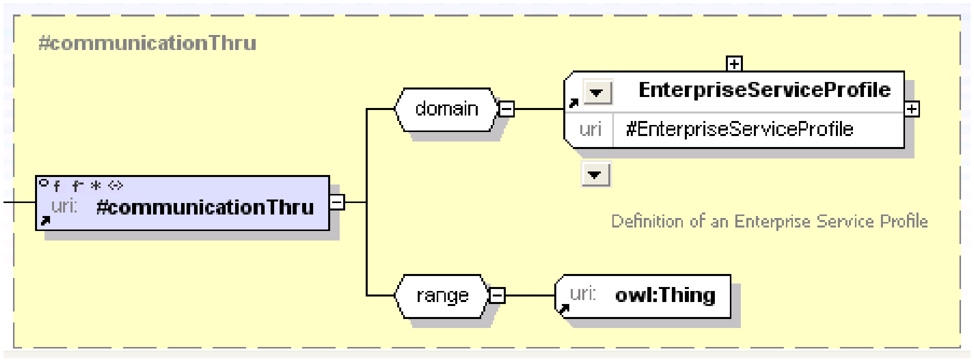


Figure 10. Enterprise Service Profile attributes - Communication Through

**Attribute Name:** Transaction Guarantee
**Description:** Non-trivial services can involve simple or complex transactions. The guarantee pertaining to the transaction has to be stated by the service provider, as this is an important criterion for service selection based on client’s transaction and security requirements. More details regarding service transactions are listed in the Service Process Model Ontology.
**Rationale:** This attribute has been added to support service selection based on consumer’s transaction guarantee requirements.

**Attribute Name:** Quality Rating
**Definition:** Service ranking
**Description:** This attribute signifies the quality rating achieved by the service. Similar to the communication through attribute the range for quality rating has not been restricted, thus allowing the uses to employ their custom or a well-established ontology for this purpose.
**Rationale:** This property has been included in Enterprise Services Profile to cater for quality requirement criteria, which is a basis for selection of services. This property can also refer to a custom ontology or a globally accepted ontology for quality ratings

### *B. Ontologies & Description Languages Applied*

The Enterprise Service Profile ontology described above builds upon a number of other ontologies along with syntax from RDF & OWL-S. These are given inTable II.

TABLE II.
ONTOLOGIES USED FOR BUILDING ENTERPRISE SERVICE ONTOLOGY (ESO)

| | Ontology /Language Name | Description | URI |
|---|---|---|---|
| 1. | NRL Security | Security Ontology for Annotating Resources | http://chacs.nrl.navy.mil/projects/4SEA/NRLOntologyFiles/securityMain.owl |
| 2. | Service | W3C Ontology for describing services | http://www.daml.org/services/owl-s/1.1/Service.owl |
| 3. | Process | W3C Ontology for describing process | http://www.daml.org/services/owl-s/1.1/Process.owl |
| 4. | Profile | Ontology to describe the profile exposed by a service | http://www.daml.org/services/owl-s/1.1/Profile.owl |
| 5. | Actor | Ontology for describing | http://www.daml.org/services/owl-s/1.1/ActorDefault.ow |
| 6. | Country | Static list of countries in OWL notation | http://www.daml.org/services/owl-s/1.1/Country.owl |
| 7. | RDF | Resource Description Framework | http://www.w3.org/1999/02/22-rdf-syntax-ns |
| 8. | RDF Schema | Adds data type to RDF | http://www.w3.org/2000/01/rdf-schema |

### C. *Enterprise Service Ontology*

To incorporate the Enterprise Service Profile the Service ontology is extended by introducing the new Enterprise Service Ontology. This ontology extends the Service ontology and provides a placeholder for holding metadata information. By building on the existing Service ontology we can still continue to make use of Service Model and Service Grounding which are mature ontologies that are relevant in the Enterprise Services environment as well.

This section discusses the Enterprise Service Ontology design and the incorporation of the Enterprise Service Profile within it. Figure 11 shows the OWL definition for our Enterprise Service, extending from the Service class. In the subsequent section, the above illustrated framework will be paired with a mechanism to persist and disseminate Enterprise Profile information.

```
<!--Enterprise Service -->
<owl:Classrdf:ID="EnterpriseService">
            <rdfs:label>EnterpriseService</rdfs:label>
            <rdfs:comment>See comments
above</rdfs:comment>
</owl:Class>
<owl:Classrdf:about="#EnterpriseService">
            <rdfs:subClassOfrdf:resource="&service;Service"/>
</owl:Class>
```

Figure 11. Enterprise Service Ontology XML Description

Similar to the Service ontology the relationship between EnterpriseService&EnterpriseServiceProfile is established by defining the former as an attribute (presents) of the latter as shown in Figure 12 and Figure 13. Another point to note is there is no restriction on the number of profiles a service presents.

### *D. Framework Components*

The proposed framework for metadata management is a combination of three elements. The first component of this framework is the Enterprise Service and Service Profile ontologies. These provide the schema that is to be used for persisting metadata. In the previous section, details of its attributes for these ontologies are discussed in detail. The other component of the Enterprise Metadata Management Framework are the mechanism that is prescribed for using the framework and the last component is the design of the Semantic Object Model and a registry repository that has been employed in building the proof-of-concept semantic metadata repository.

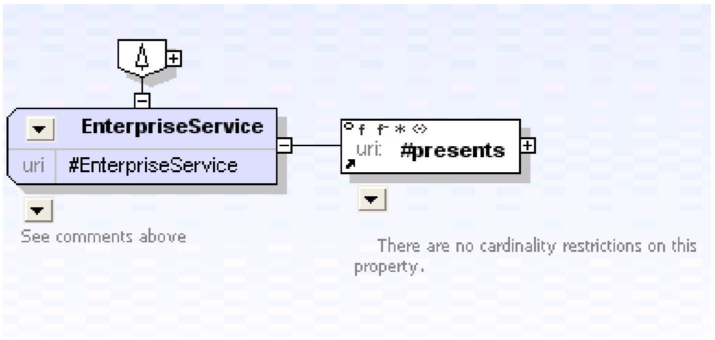


Figure 12. Enterprise Service Ontology

```
<!--  Presenting a profile    -->
<owl:ObjectPropertyrdf:ID="presents">
        <rdfs:comment>
                 There are no cardinality restrictions on this property.
        </rdfs:comment>
        <rdfs:domainrdf:resource="#EnterpriseService"/>
        <rdfs:rangerdf:resource="&eprofile;#EnterpriseServiceProfile"/>
        <owl:inverseOfrdf:resource="#presentedBy"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="presentedBy">
        <rdfs:comment>
                There are no cardinality restrictions on this property.
        </rdfs:comment>
        <rdfs:domainrdf:resource="&eprofile;#EnterpriseServiceProfile"/
>
        <rdfs:rangerdf:resource="#EnterpriseService"/>
        <owl:inverseOfrdf:resource="#presents"/>
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID="isPresentedBy">
        <rdfs:comment>deprecated form</rdfs:comment>
        <owl:equivalentPropertyrdf:resource="#presentedBy"/>
</owl:ObjectProperty>
```

Figure 13. Enterprise Service Ontology - Attribute Definition

## IV. RDF/OWL-S BASED ENTERPRISE METADATA REPOSITORY DESIGN

Service Repositories are the accepted mechanism of storing metadata information in a SOA. Both UDDI and ebXML provide limited ability to store metadata pertaining to services that are registered with them [7, 8, 13]. To validate that Enterprise Service Ontology described in the previous section we implement a repository that uses this ontology as a schema to store metadata for web services.

Extending the Service ontology to incorporate required characteristic for Enterprise level services provides the necessary foundation in the form of a framework that can be used to construct an Enterprise Service Metadata Repository (ESR). Here we detail the process with design specifications that was used to build upon the previously discussed Enterprise Service & Enterprise Service Profile ontologies a functioning proof-of –concept. The resulting product fulfills the two core functions of a metadata repository; namely to persist and disseminate service metadata and provides a semantic object model that can be used to build integration with other components in an SOA.

## *A. Use Cases for Enterprise Service Repository*

The proposed repository will have essentially three main objectives:

- Persisting service metadata
- Disseminating service metadata
- Access metadata semantic object model (MSOM)

Table III through Table V detail use cases pertaining to these objectives. To better illustrate the use case,Figure 14 shows the visual representations of the use cases.

TABLE III.
USE CASE 1 - PERSISTING SERVICE METADATA

| | |
|---|---|
| **Use Case Name:** | Persisting service metadata |
| **Actors:** | Service Developer/Provider |
| **Description:** | The service provider stores the ontology of a newly created service in the enterprise service repository. This will allow repository users to search for services based using the metadata defined in service ontologies. The service provider gives the location of the enterprise service, profile, model and grounding ontology. The repository reads the files and stores them according to their relationship using URI as unique identifies. |
| **Preconditions** | User has privileges to update the service repository<br>Ontology files have proper unique resource identifies and are well formed. |
| **Post conditions:** | Service metadata is available for retrieval and querying. |
| **Normal Flow:** | Service provider selects the option to register a new service with the repository.<br>Enterprise Service Repository (ESR) prompts the user for location of ontology files.<br>User enters the location of files and selects register.<br>Repository verifies the ontology files.<br>Verified (well-formed) ontologies are stored and user is notified with that the operation is successful. |
| **Alternative Flow:** | None |
| **Exceptions:** | Ontology Verification Fails<br>If the ontology verification fails at step 4, the user is notified of regarding the failure and the operation is aborted. |

TABLE IV.
USE CASE 2 - DISSEMINATING SERVICE METADATA

| | |
|---|---|
| **Use Case Name:** | Disseminating service metadata |
| **Actors:** | Service consumer |
| **Description:** | Service consumer requests Enterprise Service Ontology using the URI for a service previously registered with the repository. ESR matches the URI to stored ontologies and serves it to the customer. |
| **Preconditions** | Service consumer has privileges to access the service repository.<br>Service consumer has the URI for the required ontology. |
| **Post conditions:** | None. |
| **Normal Flow:** | Service consumer selects the option to retrieve specific ontology related with a service.<br>Enterprise Service Repository (ESR) prompts the user for appropriate URI.<br>ESR matches the URI to a stored ontology.<br>ESR serves the required ontology. |
| **Alternative Flow:** | None |
| **Exceptions:** | URI Match Fails<br>URI provided by the user does not match any stored ontology and ESR returns an error. |

TABLE V.
USE CASE 3 - ACCESS METADATA SEMANTIC OBJECT MODEL

| | |
|---|---|
| **Use Case Name:** | Access metadata semantic object model |
| **Actors:** | Service Developer/Provider & Service Consumer |
| **Description:** | Service developer or consumer requests access to service semantic object model. |
| **Preconditions** | 1. Service consumer has privileges to access the service repository. |
| **Post conditions:** | None. |
| **Normal Flow:** | 1. Service developer or consumer requests access to semantic object model by providing a URI or Ontology Id.<br>2. Repository parses the ontology and converts to in memory object model.<br>3. Semantic is exposed for read only or read/write access (either by RPC or RMI). |
| **Alternative Flow:** | None |
| **Exceptions:** | URI Match Fails<br>1. URI provided by the user does not match any stored ontology and ESR returns an error. |

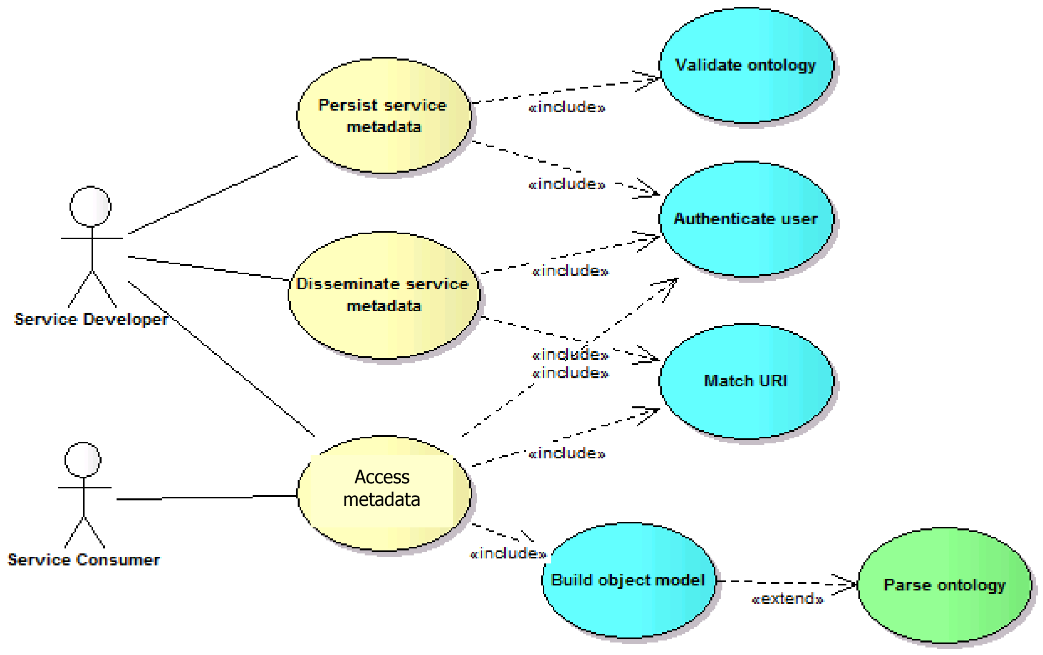


Figure 14. Use Cases for Enterprise Service Repository

## B. Domain Model for ESR Ontologies

Using the information in use cases given in Table III to Table V, we depict a domain model for our ontology repository. The objective is to retain all necessary information of the domain, so that it is easy to comprehend and useful in defining the remaining aspects of the system functionality. Furthermore using this apparatus, we will relate the objects in the system domain to one another and define important concepts and terms.

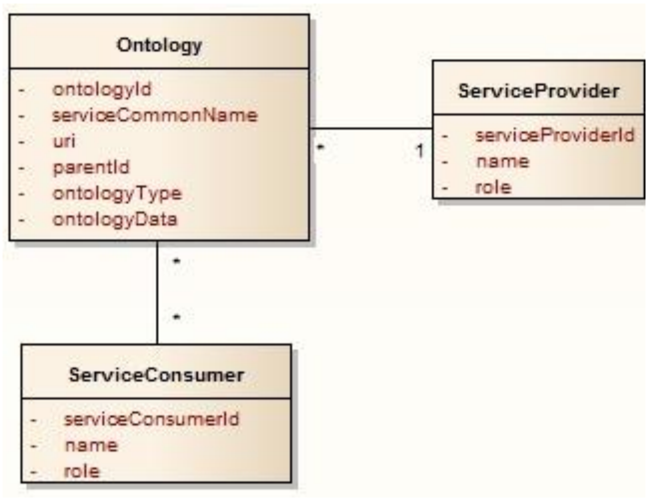


Figure 15. Domain Model for ESR Ontologies

The domain model given in Figure 15 shows the main objects in our proposed enterprise domain. The objective of this model is not to give an all-encompassing design, but rather highlight the most pertinent elements. From first view it might be perceived that the Service Producer and Consumer and similar but they have different relationship with the Ontology class, furthermore on detailed analysis further attributes will become apparent that will differentiate these two classes.

Attributes of the ontology class that warrant attention are URI and ontologyType. URI is used to uniquely identify ontologies and type is one of the four possible types. parentId is employed to identify the Container ontology for service profile, model and grounding ontologies. The information regarding the child ontologies is not persisted in the parent because it is contained in the serialized XML format stored as ontologyData attribute.

### B.1. Metadata Semantic Object Model

In order to expose service metadata we create a metadata semantic object model (MSOM). The abstract structure of this data model is shown in Figure 16. The actual object model at runtime differ as inner classes are generated from paring the stored ontology and using the class definitions and annotation in the Enterprise Profile, Model, Ground and Process.

The MSOM has been introduced to the design of the Enterprise Service Registry as it provides a convenient mechanism to interact, modify and serialize the ontology. Furthermore, for some clients the interaction with the service using RPC or RMI provides a performance edge over simple XML messaging.

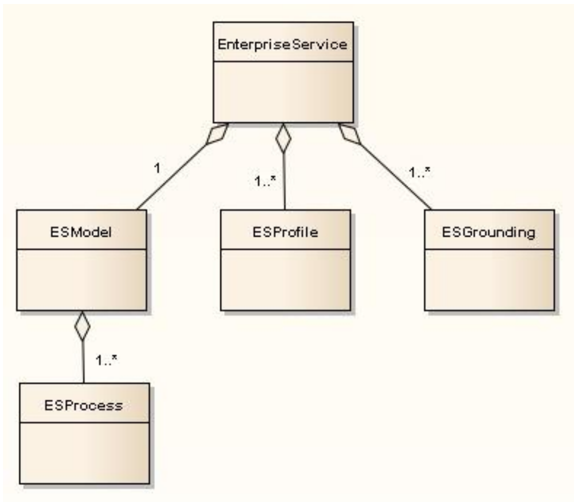


Figure 16. Metadata Semantic Object Model

*C. Sequence Diagram*

The objective of including sequence diagrams in to show the interaction of various objects in the system and the duration of their lifetime. Each sequence diagram provides an overview of the communication required to accomplish single use case. The sequence diagram in Figure 17 corresponds to the use cases inTable III and Table IV is given here along with relevant elucidation.

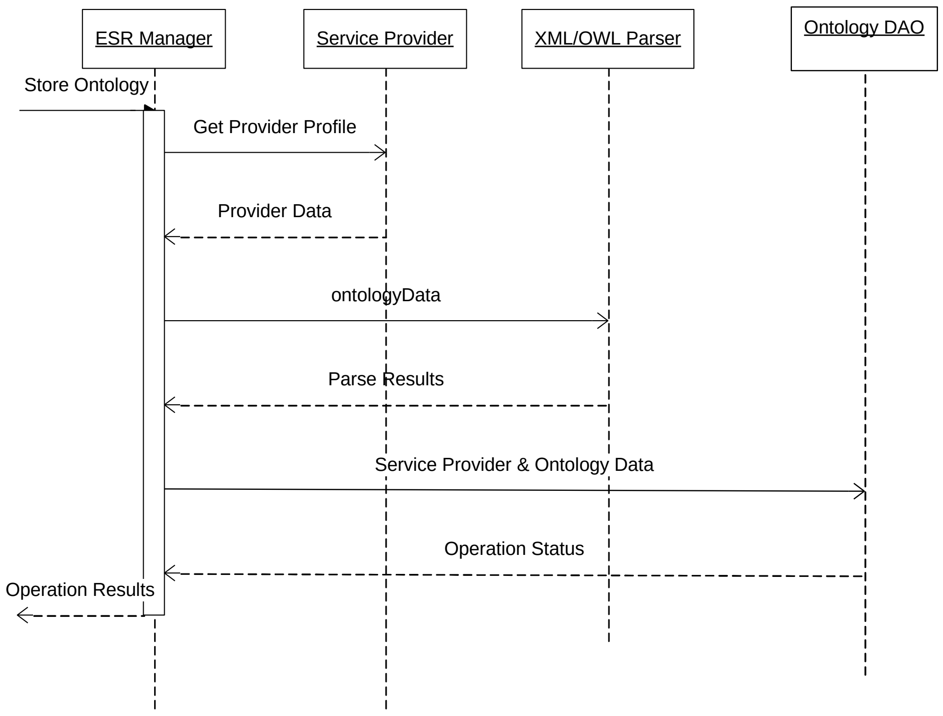


Figure 17. Sequence Diagram (Persisting Service Ontology)

The front-end object in this sequence in Figure 17 handling incoming and outgoing communication for the repository is termed as the ESR Manager. This object will be primarily responsible for implementing this use case working with other entities depicted in the domain model. The ESR Manager receives the message that includes Service Provider identification along with the XML files (Service Ontology) to be stored; it parses the files using XML/OWL parsers and on successful completion forwards the data to the Ontology Data Access Object to store to the database. Once this transaction is successful, the Service Provider is notified by means of a return message.

*D. Activity Diagram*

Activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams from structured development. The objective of including Figure 18 is two folds; firstly, to illustrate the flow logic of the corresponding use case, secondly to relate various responsibilities to objects identified in the domain model (Figure 15) and the sequence diagram (Figure 17). The main features of note within this activity diagram are that responsibility of controlling the entire use case is handled by the ESR Manager object and the use case may terminate prematurely if either the ontology is not valid or provider profile is not available. On further detailed design, ESR Manager may be decomposed into multiple objects that corresponds to fulfill ESR Manager’s tasks.

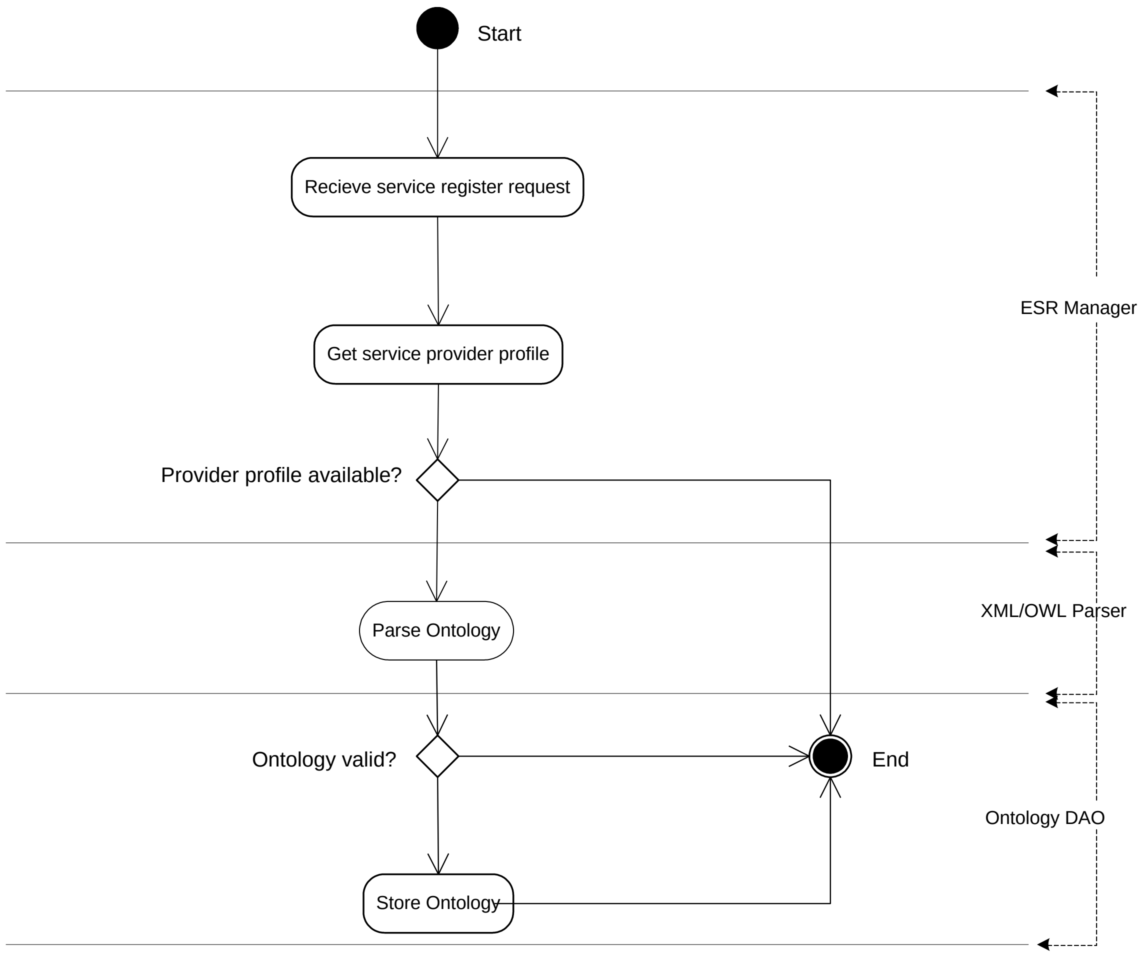


Figure 18. Activity Diagram (Persisting Service Ontology)

### E. *ESR Architecture*

It is essential to emphasize significant architectural aspects of our proposed Enterprise Service Repository to fully comprehend the end product.

The ESR is based on a layered architecture. The foundation is provided by a server based runtime environment, facilitating application development, this can be either .NET framework or Java Enterprise Edition. The ESR core relies on XML and OWL parsers to parse incoming and outgoing ontology data. Furthermore, all interaction with the persistence system is facilitated through the DMBS connectivity layer that abstracts the physical implementation of the DBMS from the class model used by the ESR Core. ESR Architecture is shown in Figure 19.

### F. *Implementation Approach & Technologies*

The implementation approach for our proposed Enterprise Service Repository was developed keeping in mind the following objectives.

- Modular design – to foster change and shield from its impact
- End products should support and comply with existing standards (HTTP, SSL, XML, WSDL etc.)
- Use open source software and frameworks (XML & OWL parsers, RDBMS, Development Platform).

#### *F.1. Implementation Approach*

To develop our proof-of-concept Enterprise Services Repository for Service Ontologies we have decided on a modular approach to development, as the domain model described earlier although sufficient for the first implementation is in no way complete. Future addition of objects and attributes is expected; therefore, any approach should cater for change with minimal impact on the existing ESR implementation.

To achieve the goal of modularity the ESR core will be implemented independent of the interactions with the various parses and the database. This will be achieved by means of an interface layer, for dealing with XML and Ontology parses and the Database.

The service consumers and providers will interact with the ESR by means of Hyper Text Transfer Protocol HTTP. This is the most prevalent and wide adopted protocol for communication on the web, ensuring communication with large segments users within the Enterprise Ecosystem and outside. Another advantage of using this protocol is that necessary required underpinnings for its use are already available in most Application Development Platforms.

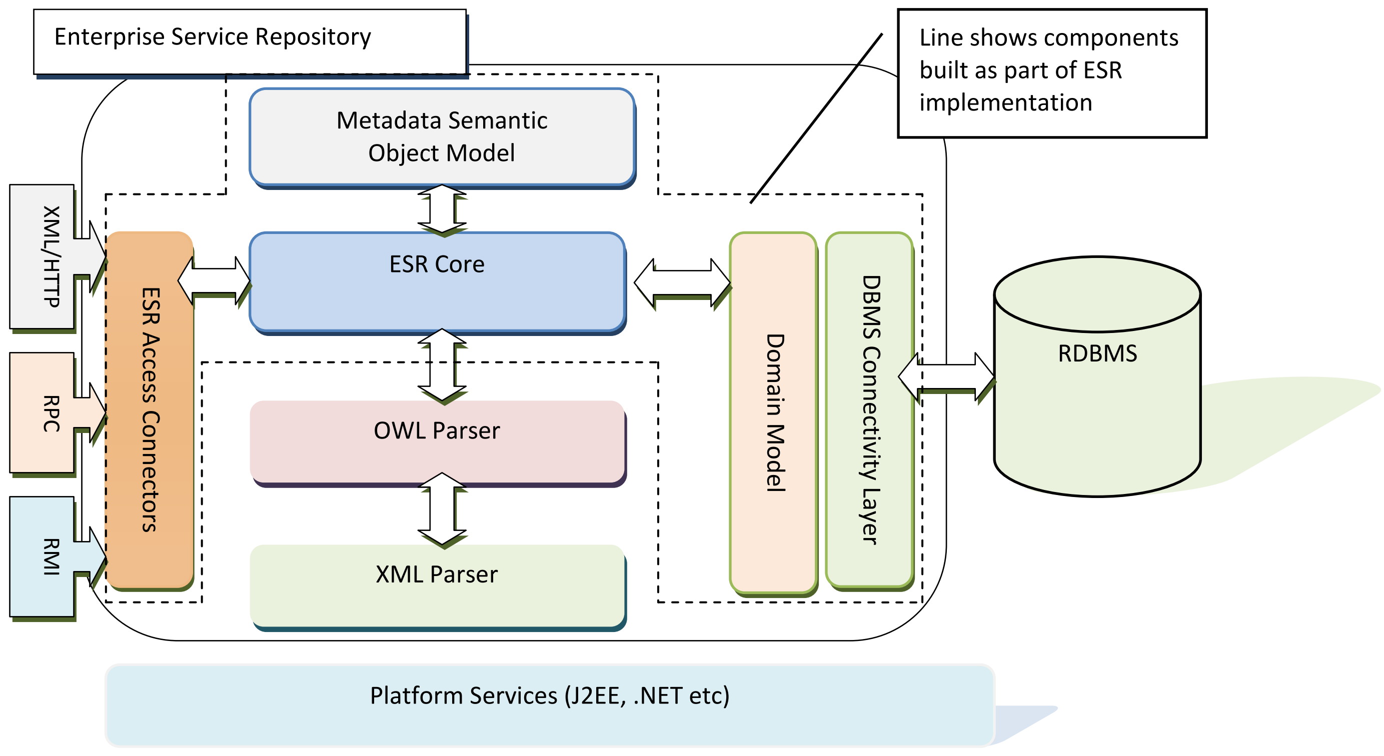

Enterprise Service Repository
Line shows components built as part of ESR implementation
Metadata Semantic Object Model
XML/HTTP
RPC
RMI
ESR Access Connectors
ESR Core
OWL Parser
XML Parser
Domain Model
DBMS Connectivity Layer
RDBMS
Platform Services (J2EE, .NET etc)

Figure 19. ESR Architecture components and interactions

## V. Results & Findings

Adhering to the proposed design outlined for a RDF/OWL-S based Semantic Metadata Repository a functioning proof-of-concept repository was developed. This repository satisfies the design goals and requirements set forth for an enterprise metadata management framework and proves its viability. The implementation of this ESR has purposefully been kept austere as the main objective of this work was to provide the framework for metadata management and the semantic repository is used as means to that end. Important findings and observations made during this exercise are given in the subsequent sections.

### *A. Working of Semantic Metadata Repository*

The end product of this exercise is enterpriser service repository (ESR) that can be used to store metadata pertaining to services and support the development and governance of services in an Enterprise Services environment. Working of the main aspects of the ESR is explained here by depicting the interaction between participating system components.

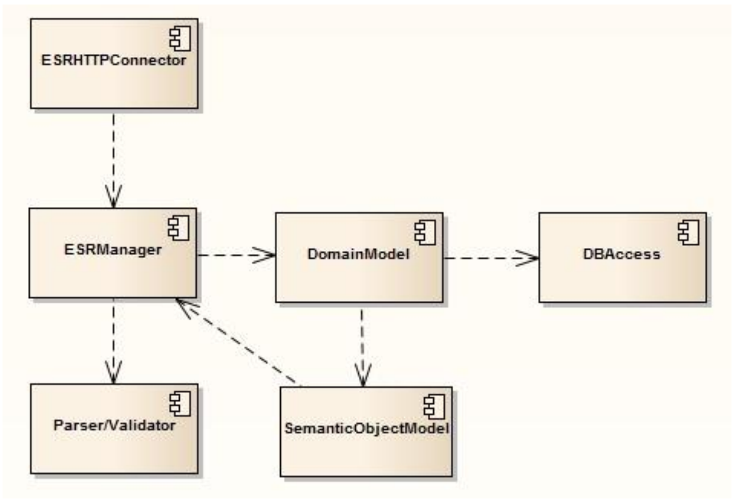


Figure 20.ESR Component Model

The ESR has been developed on top of a J2EE Web (Apache Tomcat) server that provides the typical services that are required for interaction with clients using HTTP protocol. Implementation details of the components for the ESR shown in Figure 20 and their working are given below.

1. ESR HTTP Connector - these are essentially Java HTTP Servlets that handles all incoming requests to the ESR by clients. The Servlets that are invoked depend on the type of request that is to persist service profile or perform some form of query on the repository (as described in use cases).
2. ESR Manager – this is the controller for the entire repository. It has been implemented as a Java class and it choreographs the parsing, storage, retrieval of semantic object model for OWL-S ontologies stored in the system.
3. Parser/Validator – set of XML, OWL-S and SPARQL parsers
4. Domain Model – provides an abstract layer on top the DB persistence components and handles object-relation mapping.
5. Semantic Object Model – represents an in-memory model of the service metadata stored in the DB. This model can be used for queries and modification to the service profile, grounding, process or model.
6. DB Access – these are utility classes that handle the interaction with the underlying relational database.

## *B. ESR Position in Enterprise SOA*

An Enterprise Service-Oriented Architecture is based on a number of components, such as service bus, middleware, mediation service, registries and repositories. These work together for proper functioning and delivery of services. This framework for enterprise service metadata management and the subsequent proof-of-concept repository fits within the SOA infrastructure, without any impetus mismatch with existing components. The positing of this ERS is shown in Figure 21.

To integrate with existing web service registries we rely on the works of Paolucci et al. [38, 39], practical implementation of which has been demonstrated in the form of OWL-S 2 UDDI converter. This tool supports the conversion of OWL-S profile to be advertised within a UDDI.

Another important aspect of a service lifecycle is the phases prior to its deployment. These include inception, design, development and testing. Traditionally web service registries and repositories have been only used to store information about production ready services, but with a repository built on semantic web standards support for complete service lifecycle can be provided, from inception to post production support. Therefore, the ESR provides enhancements to SOA governance and to service life cycle management.

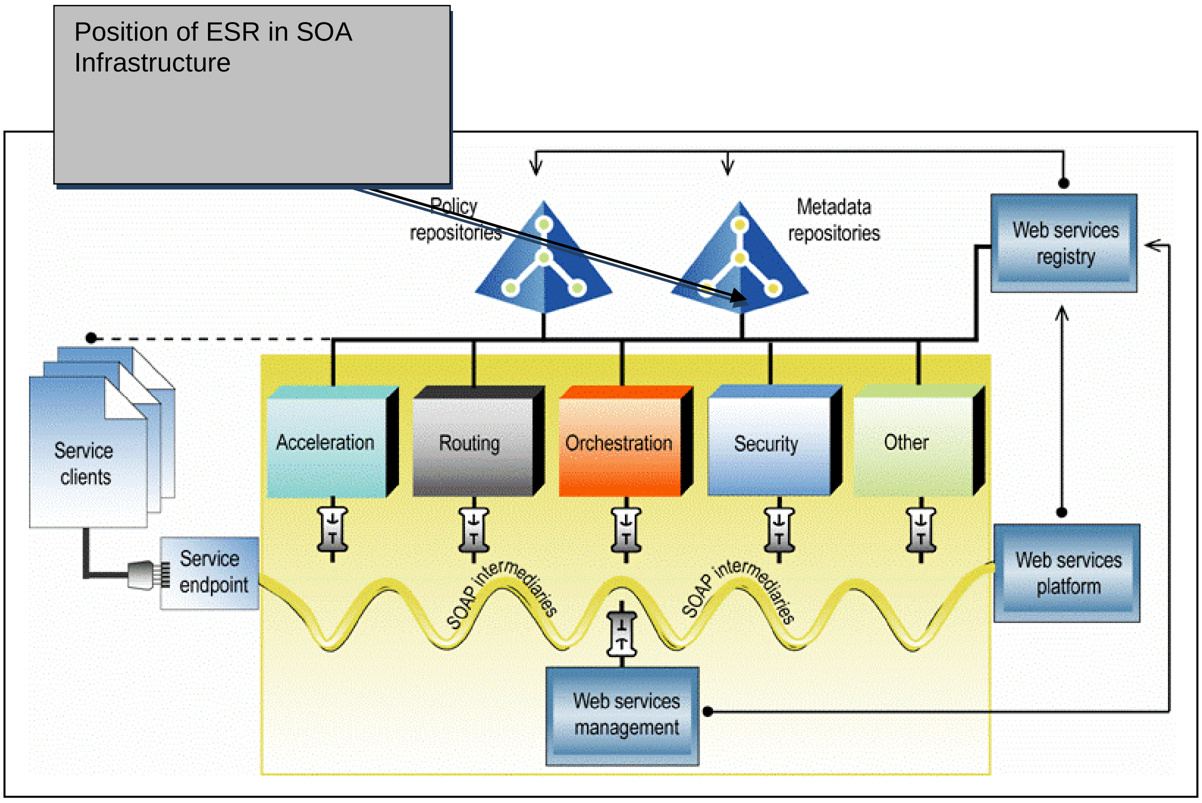


Figure 21. ESR position in Enterprise Services environment

### *C. Registry Repository Feature Comparison*

The predominate registry/repository are UDDI and ebXML.

TABLE VI.
REGISTRY/REPOSITORY COMPARISON MATRIX

| Category/ Feature | ebXML | UDDI | Semantic ESR |
|---|---|---|---|
| Service Description Standard | WSDL 1.1 | WSDL 1.1 | WSDL 1.1, OWL-S 1.0 |
| Object-Oriented API | Yes | No | Yes. API to access metadata stored in ontologies as objects |
| Object-Oriented Information Model | Yes | No | Yes. Provides semantic object model |
| Extensible API | Yes | No | Yes. Based on open standards can be extended to provide additional features |
| Registry | Yes | Yes | No. Can be used in conjunction with UDDI or ebXML registry |
| Repository | Yes Integrated registry-repository Any type of electronic content supported | No | Yes. Supports internal storage or links to external electronic resources. |
| Predefined queries | Yes | Yes | Yes |
| User-defined queries | Yes | No | Yes. User can define SPARQL queries |
| Ad hoc queries | Yes | Yes | Yes |
| SQL query syntax | Yes | No | No. Support for SQL can be provided by allowing direct access to OWL models stored in the ESR database |
| SPARQL query syntax | No | No | Yes |
| Semantic query | No | No | Yes |
| XML query syntax | Yes | Yes | Yes |
| Predefined taxonomies | Yes | Yes | Yes |
| User-defined taxonomies | Yes | Yes | Yes. Through extension of Enterprise Service Profile |
| Classification of artifacts | Yes | Yes | Yes |
| Classification of metadata objects | Yes | No | Yes |
| User Defined Security | Yes | No | Yes. Supports user defined fine grained security via service profile. |

From Table VI, we can note the following important points of comparison between, ebXML, UDDI and Semantic ESR:

1. SQL query support – although it is not currently supported by ESR this capability can easily be introduced as ontologies are persisted in a relational database.
2. Registry support – this feature can be added to the ESR by either building a registry module or integrating with an existing registry, such as UDDI.
3. SPARQL, semantic queries – these are only supported by ESR allowing data to be searched based on semantics specified by the clients

### *D. Advantages of Semantic ESR*

Although tangible benefits and advantages of a RDF/OWL-S based repository can truly be realized after in field use and testing, still merger of Enterprise Service Profile ontology with traditional repository functions offers many advantages. Some of the most apparent and unique of these advantages are given below.

#### *D.1. Service Discovery*

Registry repositories provide the capability to search for registered services. This search often involves human intervention and is usually limited to searching within the limited registered info pertaining to the service. Enterprise Service ontology supports search by automated agents as well as humans. Furthermore, the search is not limited just to syntax but also supports semantics.

#### *D.2. Service Reuse*

Semantic ESR has been designed to maximize service reuse, by providing a very flexible search mechanism (SPARQL) and supporting service life cycle management. As the service ontology contains not only profile information, but also model, process and grounding more detailed searches are possible, promoting service reuses.

#### *D.3. Service Governance*

Marinating and monitoring an Enterprise SOA is another aspect where ESR can provide many advantages such as:

1. Service dependencies are registered within the service profile. Providing information about critical service within the enterprise architecture. ebXML and UDDI do not provide an explicit capability for this purpose.
2. Detailed service security requirements are captured in the service profile. Supporting service selection to be done on the basis of security requirements of the consumer.
3. Support for persisting and managing all electronic artifacts related to a service. This feature is supported by ebXML but not by UDDI.

## VI. CONCLUSION AND FUTURE WORK

Service-Oriented Computing and Service-Oriented Architecture have become an integral part of many organizations. Large businesses rely on these technologies to run their day-to-day operations. They also gain a competitive advantage by leveraging existing investment in services by reuse, thereby reducing time-to-market of new products and services. Along with these new technologies comes the challenge of complexity of services infrastructure and it governance. Enterprise registry & repositories represent an effective of tackling these growing pains of an SOA. The focus of this work has been to improve the capability of repositories by capturing service metadata using semantic web standards of RDF & OWL.

In this research, we proposed an Enterprise Service ontology and Enterprise Service Profile ontology. This is essentially an extension of the Service ontology [34], modified to capture metadata of an enterprise web service. The Enterprise Service Profile caters for capture of metadata attributes, which are significant in supporting service discovery, reuse and governance. These include service provider info, access role, security requirements, runtime platform, dependencies and quality ratings to name a few. The Enterprise Service Profile does not restrict the user to only these metadata elements, yet it provides typical attributes of interest in most organizations. Domain specific customized profiles can be created by extending the Enterprise Service Profile.

We also designed and implemented a rudimentary repository built on the proposed metadata management framework. This includes UML design artifacts (Use cases, sequence, activity and object diagrams) and overall architectural pattern. The findings from implementing the proof-of-concept are also detailed along with a comparison with UDDI and ebXML.

Building semantic aware applications is a new and exciting field of research and we have tried to push its boundaries with this effort. Still much remains to be done if semantic web technologies are to realize their full potential in the enterprise. The future direction of this research include: study of other constituents (model, process and ground) of service ontology to incorporate enterprise related metadata. We also plan to integrate semantic ESR with service development tools to address the capture of metadata from inception to delivery, with minimal developer intervention and use of automated agents to update service profile based on events in the SOA.

## ACKNOWLEDGEMENT

The authors acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this work.

## REFERENCES

[1] A. Arsanjani, B. Hailpern, J. Martin, and L. Tarr, "Web Services: Promises & Compromise," 2002.

[2] D. Linthicum, *Enterprise Application Integration*: Addison-Wesley Professional, 2000.

[3] D. Georgakopoulous, H. Schuster, A. Chichocki, and D. Baker, "Managing process and service fusion in virtual enterprises," *Information Systems, ACM,* vol. 24, pp. 429-456, 1999.

[4] L. M. Chamarinha-Matos, H. Afsarmanesh, C. Gartia, and C. Lima, "Towards an architecture for virtual enterprises," *Journal of Intelligent Manufacturing, Springer Netherlands,* vol. 9, pp. 189-199, 2004.

[5] P. M. Papazoglou and D. Geogakopoulos, "Service Oriented Computing," *Communications of the ACM,* vol. 46, pp. 25-28, 2003.

[6] M. P. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," in *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, 2003, pp. 3-12.

[7] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling The Web Services Web: An Introduction to SOAP, WSDL, and UDDI," *Internet Computing, IEEE,* vol. 6, pp. 86-93, 2002.

[8] C. Peltz, "Web services orchestration and choreography," *Computer* vol. 36, pp. 46-52, 2003.

[9] G. Alonso, *Web Services: Concepts, Architecture and Applications*: New York: Springer, 2004.

[10] M. Burstein, C. Bussler, T. Finin, M. N. Huhns, M. Paolucci, A. P. Sheth, S. Williams, and M. Zaremba, "A semantic Web services architecture," *IEEE Internet Computing, IEEE,* vol. 9, pp. 72-81, 2005.

[11] O. S. w. site. *UDDI Specifications*. Available: http://www.oasis-open.org/specs/index.php#uddiv3.0.2

[12] A. Hibner and K. Zielinski, "Semantic-based Dynamic Service Composition and Adaptation," in *IEEE Congress on Services*, 2007, pp. 213-220.

[13] O. S. w. site. *ebXML Registry Services Specification*. Available: http://www.oasis-open.org/specs/index.php#ebxmlrimv3.0.

[14] B. Hofreiter, C. Huemer, and W. Klas, "ebXML: status, research issues, and obstacles," in *Proceedings. Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems*, 2002, pp. 7-16.

[15] M. MacKenzie, K. Laskey, F. McCabe, P. Brown, and R. Hamilton, "Reference Model for Service Oriented Architecture 1.0," 2006.

[16] C. Zhou, P. Liu, E. Kahan, N. Wang, and Z. Xue, "Context Aware Service Policy Orchestration," in *IEEE International Conference on Web Services, ICWS* 2007, pp. 936-943.

[17] L. An and J.-J. Jeng, "Business-Driven SOA Solution Development," in *IEEE International Conference on e-Business Engineering, ICEBE 2007*, 2007, pp. 439-444.

[18] D. Nickull, L. Reitman, J. Ward, and J. Wilber, "Service-Oriented Architecture Technical Whitepaper," 2005.

[19] E. Newcomer, *Understanding Web Services: XML, WSDL, SOAP and UDDI*: Addison-Wesley, 2002.

[20] A. ShaikhAli, O. F. Rana, R. Al-Ali, and D. W. Walker, "UDDIe: an extended registry for Web services," in *Symposium on Applications and the Internet Workshop*, 2003, pp. 85-89.

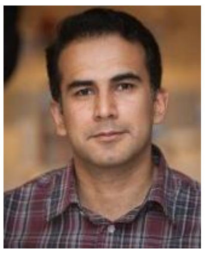
[21] J. Liu, J. Liu, and L. Chao, "Design and Implementation of an Extended UDDI Registration Center for Web Service Graph," in *IEEE International Conference on Web Services, ICWS,* 2007, pp. 1174-1175.

[22] B. W. O. S.C. Oh, E.J. Larson, and D.W. Lee. BF "Web Services Discovery and Composition as Graph Search Problem". ," in *Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE),* Hong Kong, China, 2005.

[23] K.-H. Lee, M.-y. Lee, Y.-Y. Hwang, and K.-C. Lee, "A Framework for XML Web Services Retrieval with Ranking," in *International Conference on Multimedia and Ubiquitous Engineering, MUE '07*, 2007, pp. 773-778.

[24] A. Mani, Nagarajan A., "Understanding quality of service of Web services," 2002.

[25] J. Liu, N. Gu, Y. Zong, Z. Ding, S. Zhang, and Q. Zhang, "Service registration and discovery in a domain-oriented UDDI registry," in *The Fifth International Conference on Computer and Information Technology, CIT*, 2005, pp. 276-282.
[26] A. Dogac, Y. Kabak, and G. B. Laleci, "Enriching ebXML registries with OWL ontologies for efficient service discovery," in *Proceedings. 14th International Workshop Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications*, 2004, pp. 69-76.
[27] Y. Roh, H. Kim, H. S. Kim, M. H. Kim, and J. H. Son, "Semantic Business Registry Information Model," in *International Conference on Convergence Information Technology*, 2007, pp. 2142-2145.
[28] J. Luo, B. Montrose, A. Kim, A. Khashnobish, and M. Kang, "Adding OWL-S Support to the Existing UDDI Infrastructure," in *International Conference on Web Services, ICWS '06*, 2006, pp. 153-162.
[29] Z. Feng, R. Peng, B. Li, K. He, C. Wang, J. Wang, and C. Zeng, "A Service Registry Meta-model Framework for Interoperability," in *2011 Tenth International Symposium on Autonomous Decentralized Systems*, 2011, pp. 389-398.
[30] T. Baker, D. Lamb, A. Taleb-Bendiab, and D. Al-Jumeily, "Facilitating Semantic Adaptation of Web Services at Runtime Using a Meta-Data Layer," in *Developments in E-systems Engineering (DESE)*, 2010, pp. 231 - 236
[31] R. D. Virgilio, "Meta-Modeling of SemanticWeb Services," in *2010 IEEE International Conference on Services Computing*, 2010, pp. 162-169.
[32] J. Colgrave, R. Akkiraju, and R. Goodwin, "External Matching in UDDI," in *Proceedings of International Conference on Web Services (ICWS 2004)*, San Diego, California, USA, 2004.
[33] A. Ankolenkar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web Service Description for the Semantic Web," in *Proceedings of the First International Semantic Web Conference (ISWC)*, Sardinia (Italy), 2002.
[34] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, "OWL Web Ontology Language Semantics and Abstract Syntax," 2004.
[35] Y. Li, X. Yu, L. Geng, and L. Wang, "Research on Reasoning of the Dynamic Semantic Web Services Composition," in *IEEE/WIC/ACM International Conference on Web Intelligence, WI 2006*, 2006, pp. 435-441.
[36] G. C. Gannod, R. J. Brodie, and J. T. E. Timm, "An interactive approach for specifying OWL-S groundings," in *Ninth IEEE International Enterprise Computing Conference, EDOC* 2005, pp. 251-260.
[37] B. Ma and N. Xie, "From OWL-S to PNML+OWL for Semantic Web Services," in *2010 Second International Conference on Computer Modeling and Simulation*, 2010, pp. 326-328.
[38] M. Paolucci, T. P. Kawamura, and K. TR. Sycara, "Importing the Semantic Web in UDDI," in *E-Business and Semantic Web Workshop on Web Services*, 2002.
[39] N. Srinivasan, M. Paolucci, and K. Sycara, "Adding OWL-S to UDDI, implementation and throughput," in *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, 2004.



**Malik Muhammad Umar** received Bachelor of Science degree in Information and Computer Science from Hamdard University, Karachi, Pakistan in September 1999 and his MS in computer science from King Fahd university of petroleum and minerals in 2008. He worked for Ferozons (PVT) Ltd. as a Software Engineer, till 2002. He then joined Saudi Aramco in 2002 as a Technical Consultant progressing to become Technical Architect in two years, currently he works as Senior Software Engineer at SecureKey Technologies Inc in Canada. His research interests include, Service-Oriented Architecture, Design Patterns and Enterprise Application Integration.

**Mohammad Alshayeb** is an associate professor in Software Engineering at the Information and Computer Science Department, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia. He received his MS and PhD in Computer Science and certificate of Software Engineering from the University of Alabama in Huntsville. He worked as a senior researcher and Software Engineer and managed software projects in the United States and Middle East. He is a certified project manager (PMP). His research interests include Software quality, software measurement, and metrics, and empirical studies in software engineering.