Framework for Service-Oriented Architecture Metadata Management

JARALLAH ALGHAMDI, MALIK UMAR
Information & Computer Science Department
King Fahad University of Petroleum & Minerals
Dhahran, SAUDI ARABIA

Abstract: - Enterprise Architects and developers constantly have to deal with changing and evolving business requirements. Organizations have to be more dynamic in their collaboration and competition efforts to remain viable. To combat these increasing pressures on the IT resources enterprises are moving towards the paradigm of service orientation. This allows for businesses to leverage their existing investment in IT to accommodate new requirements. But this service orientation comes at the cost of increased complexity of the enterprise systems architecture. This paper explores the possibility of using semantic web technologies of XML, RDF and DAML + OIL and proposes a framework to curtail the increasing complexity of Service–Oriented Architectures.

Key-Words: - Service-Oriented Architecture, Metadata management, RDF, DAML + OIL

1 Introduction

In today's ever increasingly fast paced business environment organizations have to adapt quickly to market and global socio-political changes. Organizations that can perform this task survive to face the next set of challenges and those that cannot, go out of business. A key factor that has emerged over the years is technological improvements to the way business is done. This can greatly enhance the profitability or hinder it depending on the way technology is used. Information Technology infrastructure of any business must be agile and adaptive to the way business is being performed now and in future. The current approach of developing enterprise applications lacks this agility and cannot be easily and quickly aggregated to solve a new set of problems.

Service Oriented Architecture (SOA) attempts to solve some of these inter-business and intra-business development and integration problems. Service Oriented Architecture is a new paradigm in distributed systems aiming at building loosely-coupled systems that are extendible, flexible and fit well with existing legacy systems [1]. Organizations can build flexible architectures utilizing SOA.

The flexibility offered by SOA comes at the price of complexity of overall Enterprise Software Architecture [2]. A complex picture emerges due to the layered nature of the enterprise architecture and the dependency of one layer of services on another. Although this type of architecture is excellent for flexibility but it is difficult to maintain as services become dependent on one another and get intertwined. Configuration Management Databases [2] alleviate some of the complexity but still do not provide a dynamic overall view of the service usage and dependencies to architects. What is needed is a metadata management framework for web services that would allow storage of service dependencies and other pertinent information in an easily accessible repository. This would give enterprise architects a mechanism to manage overall complexity and reduce problems involved in any type of service maintenance.

2 Service Oriented Architecture: An Overview

To meet the ever changing requirements of the business world, software industry has developed a number of solutions to reduce the time to market. The latest trend in the information technology industry, as a part of this continuing effort, is to move to service orientation (SO) paradigm which is based solely on industry accepted open computing standards. The service orchestration paradigm advocates "services" being available as discoverable resources on the network.

A service is defined as "A contractually defined behavior that can be implemented and provided by a component for use by another component" [11]. Services abstract their internal complexity and working details from the services' consumer and provide their functionality through a contract façade. Web Services is an industry accepted standard for implementing distributed components as services. Web Services employ open interoperability standard of XML [12] and SOAP [15] to exchange data. The

Web Service contract information is defined via the Web Service Description Language (WSDL) [13] and Web Services are discoverable for use on the network by registering with repository for Universal Description and Discovery Integration (UDDI) [4-6, 14]. Fig.1 gives an example of how a Web Services Architecture works; depicting the registry-lookup and invocation processes. Web Services are the premiere technology in enabling the service-orientation of IT. The advent of web services has enabled a true implementation of Service–Oriented Architecture. The combination of these open standard technology allows Web Services to interconnect disparate systems in an object neutral way.

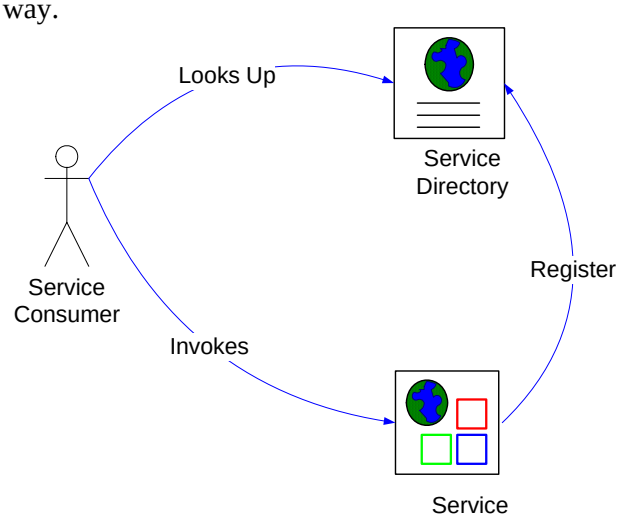

Fig.1 Web Services Architecture

There are many different definitions of SOA [16, 17]. Syed Hashimi explains in [4] that SOA software applications are built on basic components called services, processes etc., defined in terms of what the component does, typically carrying out a business-level operation. A service in SOA is an exposed piece of functionality with three properties.

1. The interface contract to the service is platform-independent.
2. The service can be dynamically located and invoked.
3. The service is self-contained. That is, the service maintains its own state.

Service-Oriented Architecture is not a new concept. There have been previous attempts at attaining services orientation such as the Common Object Request Broker Architecture (CORBA) [9] and the Distributed Component Object Model (DCOM) [10]. The main difference between the previous implementation and the current one is that open standards allow for true interoperability. For example; using a Web Services based SOA, a .NET application can invoke an IBM CICS or IBM IMS transaction on a mainframe or a J2EE application running on UNIX can invoke a BizTalk service running on Windows platform. Fig.2 shows how the Service-Oriented Architecture uses the Web Services.

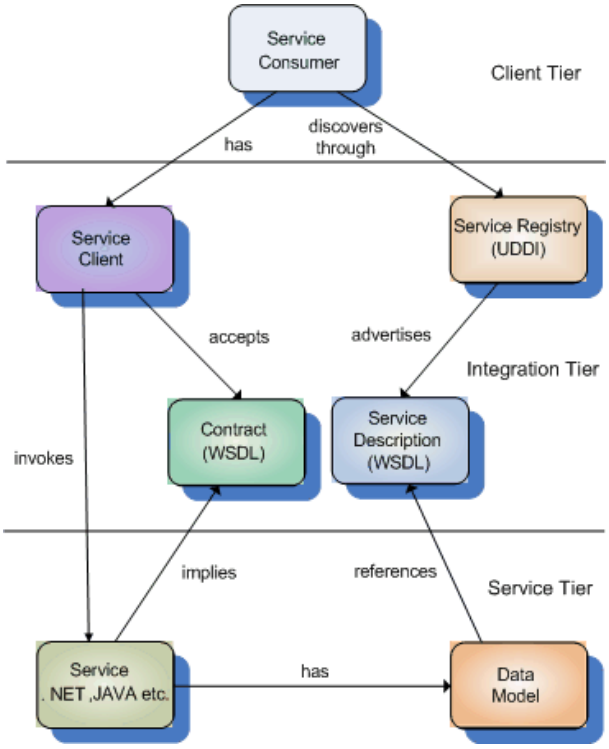

Fig.2 SOA using Web Services

3 Problem Description

With the advent of web services as a viable platform independent technology [5], 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 [4-6] and applications can quickly be built by utilizing pre-built services. New services can even be dynamically composed [7] 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 abstraction 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 impact 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

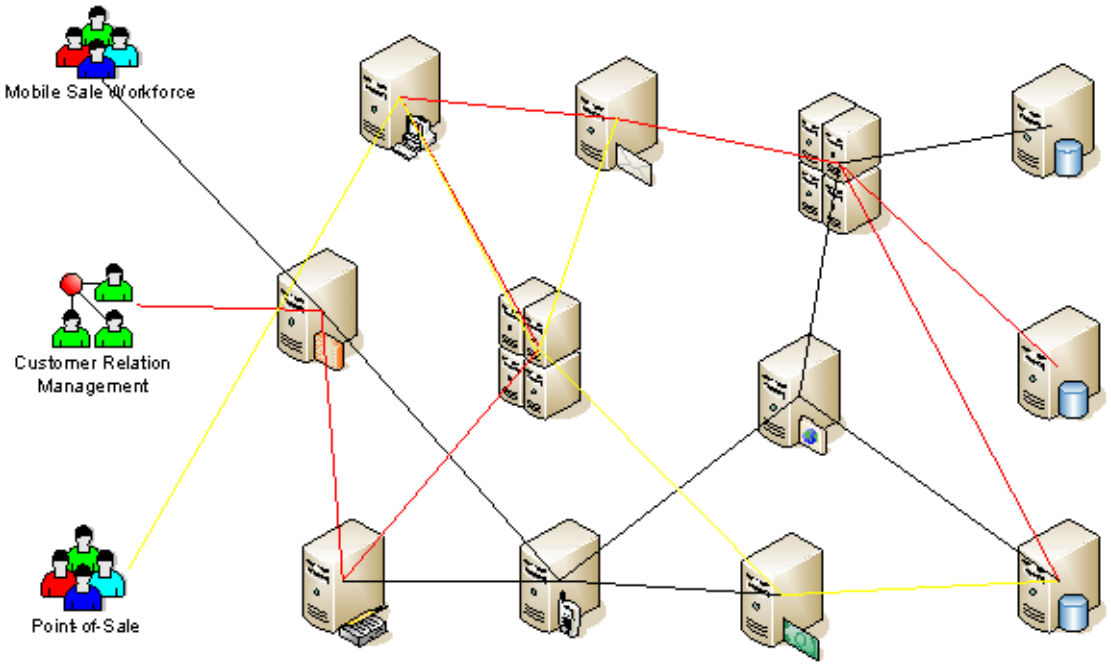

Fig.3 Enterprise Architecture Complexities

services. This dependency information becomes especially critical while architecting new solutions or maintaining existing services infrastructure. This is especially true in large enterprise architectures which are composed of a heterogeneous collection of systems interacting to provide business functionality.

Fig.3 shows the complexity of an enterprise landscape overlaid with the software relationships. Organizations continue to move to a distributed and loosely coupled architecture where an application no longer runs on a single machine but is distributed over the enterprise landscape. New generation of business software solutions are a collection of customized smaller software components, as a result the organizational IT infrastructure is growing in terms of scale, scope and complexity. This complexity is further compounded when change is introduced into the enterprise architecture due to changing business requirements, acquisitions and mergers.

Complexity cannot be eliminated in a dynamic and evolving environment so the focus has to be diverted to managing this change and complexity in order 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.

4 Related Works

There are a number of areas in which work is being done related to SOA metadata management. Some of the more pertinent items to this research work are discussed in the following lines.

Universal Description and Discovery Integration Registries provide the means for business and web service to register, for discovery and utilization by consumers. Most of the work being performed is to extend UDDIs to provide additional information about services other than service location and description [6, 18].

WSDL is used to describe the web service's interface and WSOL [19] is an XML notation compatible with WSDL standard. It has been used to provide formal specification of classes of service, service constraint and management statement for Web Services. WSOL enables selection of more appropriate Web Service and service offering for particular circumstances.

To overcome the deficiencies of XML [12] and RDF [20] schema, DARPA Agent Markup Language (DAML) [21] was developed by Defense Advanced Research Projects Agency. It can be used

to describe the semantics behinds a web service in terms of properties, constraints and relationships using the DAML+OIL [22] ontology. These markup languages overcome some of the limitations of WSDL in describing web service metadata.

Universal Description & Directory Integration registries provide an excellent and widely adopted mechanism for service providers to advertise their services in a standard form and, for service consumers to query services of interest. A UDDI entry consists of white pages; contact information, yellow pages; industrial classification and green pages; reference to specification of services. Although some of the service metadata can be stored in UDDI registries that would be helpful for consumers to select the appropriate service, yet not all information can be persisted in this manner for the following reasons.

- Requires modification of the UDDI data model.
- Requires extensions of UDDI publication interface.
- Forces service provider to ensure all UDDI registries are updated with the latest service metadata.

Therefore the metadata for a Web Services based SOA should be retained outside the UDDI registries in an independent source optimized for this purpose.

The above mentioned technologies taken in isolation do not provide complete solution to the metadata management problems for Web Services based SOA. Used in conjunction with ontology to describe this information and web based information dissemination techniques, they can provide a framework to address some of the complexity problems faced by Enterprise Architects.

5 SOA Metadata Management Framework

As we have previously mentioned apart from the basic information a service may have non-functional characteristics, such as service relations, dependencies and quality of service. Not having all the pertinent information about services available in a readily accessible manner creates complexity in a dynamic SOA. To overcome this complexity, as a first step we have identified a basic set of service attributes that are relevant for Enterprise Architects to manage and maintain the enterprise SOA. Table 1 lists a preliminary sample of service metadata & configuration information for SOA. These show some of the important aspects that architects are interested in regarding services participating in a SOA.

Name	Description	Comments/Examples
Service Description		
Service Name	Name of the business service	customerAddressService
Service Description	Detailed description of the functionality provided by the service	
Service Owner	The business proponent for the service	Customer Relations Department
Service Developer	In house or external entity responsible for developing and maintaining the web service	ACM Service Corp. CRM development Team etc
Service Category	Business service category	CRM – Customer Information
Service Scope	The scope of the business service, department, business line or enterprise wide	Enterprise Service
Service Type	Basic Service (informational) Compound Service (transactional)	Basic Service
Service Implementation Details		
Service Contract	Location of the service WSDL [13] file	http://acme.com/ws/customer?WSDL
Binding Protocol	Protocol required to access the service	HTTP
Security Protocol	Security protocol used to secure the service	HTTPS/SSL – 3
Service Controller	Controlling authority that grants access to the service	Customer Relations Dept. or CRM development Team
Service Use Case	Business requirement/use fulfilled by the service	http://acme.com/ws/customer?use_case
Class Diagram	Class diagram showing methods and attributes of the service	http://acme.com/ws/customer?use_case
Service Dependencies	Other Service on which this service is dependent for its functionality	http://acme.com/ws/authentication http://acme.com/ws/authorization

Table 1: Service metadata & configuration information for SOA

5.1 Metadata and Configuration Representational Schema

To represent the non functional characteristics of web services this paper describes a RDF and DAML+OIL based ontology that can be adopted for this purpose. The advantage of using RDF and DAML+OIL based ontology is that the resulting metadata document is in XML format that is both human and machine readable.

The RDF schema allows class definitions by declaration, for example we can define enterprise service as a class of web service which can have a unique set of characteristics, representative of this type of services. Fig.4 shows a top level RDF tag used for making a web service. Dependency information of a service can also be provided in a declarative format using RDF tags as shown in the Fig.5. This technique allows for complete hierarchy of services and their dependencies to be demonstrated, showing not only dependencies, but also weak links.

```
<rdfs:Class rdf:ID = ”WebService” >
   <rdfs:lable> Web Service
</rdf:lable>
   <rdfs:comment>Web Service part of
   organizational SOA
   </rdfs:comment>
</rdfs:Class
```

Fig.4 A Root RDF Tag

Using a combination of DAML+OIL and RDF we can define the properties of the web service class.

```
<daml:DatatypeProperty rdf:ID =
“serviceDependecy”>
   <rdfs:lable> Service Dependency
</rdf:lable>
      <rdfs:comment> Services used by
      this service
      </rdfs:comment>
      <rdfs:domain rdf:resource =
“#WebService” >
      <rdfs:type rdf:resource =
      “http://www,w3c.org/2000/10/XML
      Schema#hypertext” />
</daml:DatatypeProperty>
```

Fig.5 Dependency Information using RDF Tags

5.2 Web Service Metadata Ontology

Fig.6 describes all the referenced standards used for defining this ontology and lays the ground work with service dependencies as a sample of the framework for meta-data. Additional DAML data types can be added to this ontology similar to attributes suggested in Table. 1.

```
<?xml verion 1.0 encoding “UTF-8” ?>
<rdf:RDF
      xmlns:rdf =
“http://www.w3c.org/1999/02/22-rdf-
syntax-ns#”
      xmlns:rdf =
“http://www.w3c.org/2000/01/rdf-
schema#”
      xmlns:rdf =
“http://www.w3c.org/2001/10/daml+oil#
”
      xmlns:rdf =
“http://www.acme.com/ws/metadata”>
<daml:Ontology rdf:about = “”>
   <daml:versionInfo> 1.0
</daml:versionInfo>
   <rdfs:comment> An ontology for web
service metadata </rdfs:comment>
   <daml:imports rdf:resource =
      “http://www.w3c.org/2001/10/dam
l+oil”/>
</daml:Ontology>
<rdfs:Class rdf:ID = ”WebService” >
      <rdfs:lable> Web Service
</rdf:lable>
<rdfs:comment>Web Service part of
organizational SOA </rdfs:comment>
</rdfs:Class>
<daml:DatatypeProperty rdf:ID =
“serviceDependecy”>
      <rdfs:lable> Service Dependency
</rdf:lable>
<rdfs:comment> Services used by this
service </rdfs:comment>
      <rdfs:domain rdf:resource =
“#WebService” >
<rdfs:type rdf:resource =
“http://www,w3c.org/2000/10/XMLSchema
#hypertext” />
</daml:DatatypeProperty>
.
.
. Other data properties
</rdf:RDF>
```

Fig.6 RDF Schema

This resulting schema can be used to describe all the service metadata about the enterprise services. For example we can define the service dependencies of Customer Address Service as shown in Fig.7. It has been limited to dependency attributes and the other attributes have been excluded for brevity.

```
<?xml version = “1.0” encoding =
“UTF-8”>
<xmlns: wsMetadata =
"http://acme.com/ws/metadata.xsd">

<wsMetaData:WebService rdf:ID =
“customerAddressService”>
      <rdfs:lable> Customer Address
Service </rdfs:lable>

<wsMetaData:serviceDependency>
http://acme.com/ws/authentication
</wsMetaData:serviceDependency>

<wsMetaData:serviceDependency>
http://acme.com/ws/authorization
</wsMetaData:serviceDependency>
.
.
. Other properties
</wsMetaData:WebService>
```

Fig.7 Service Dependency Definition

5.3 Web Services Metadata Dissemination

This section describes various techniques that can be adopted to distribute web service metadata and configuration information to interested parties.

Discovery of web services is done by interrogating UDDI registries, where basic service description and location information is stored. The UDDI tModel can be used to register the service metadata file locations.

The second technique that can be used to disseminate service related metadata is to modify UDDI data model to accommodate service metadata. The advantage of this technique is that metadata regarding the service could be captured at the time of service registration. Also, the service discovery would be highly efficient. The main disadvantages of this technique are that it would require modification to the UDDI data model; requiring all service consumer lookups for services to adapt to this new model. Also, if the service characteristics change after deployment, the information in the registry would become outdated. To counter this drawback the service provider would have the additional burden of updating the registry every time any service related metadata changes.

The third technique could be to use relational databases to retain the service information. Using this technique, a UDDI tModel would provide the service consumer access to an interface that would allow interrogation of the underlying relational database. The advantages of this mechanism include no changes to the UDDI data model, efficient service discovery and dynamic updates to service metadata and configuration. The main disadvantage of this form of information dissemination is the dependency of having an RDBMS available for service configuration, creation of interfaces to store and maintain metadata for service provider and an additional interface for consumers to interrogate the RDBMS for service metadata.

Keeping in mind our original objective of managing SOA complexity in a dynamic environment, service metadata distribution apparatus most suitable for our purpose is the use of tModels to point to an external resource provided by the service itself by means of a URL. For example, our Customer Address Service’s metadata can be made accessible similar to the way WSDL files are accessed: http://acme.com/ws/customer?metadata.

6 Conclusion and Future Work

In this paper we have proposed a framework for service metadata management that can help Enterprise Architects combat complexity in their organizational SOA. The is framework provides a basic set of web service characteristics and an RDF DAML+OIL based ontology to describe these in a metadata document. Furthermore the framework also provides a means of disseminating this information utilizing the capabilities of exiting UDDI registries.

Based on the RDF DMAL+OIL languages this paper proposes a metadata management ontology for web services, basic set of service metadata properties and a service metadata distribution technique that would alleviate some of the complexity problems in Service-Oriented Architectures. This framework can provide architects visibility into service relations, dependencies and other non-functional characteristics. As service implementations change in a dynamic environment this framework would give an accurate picture of the current state of an organizations SOA.

Future work of interest includes creating a comprehensive list of service characteristics and their classification that are of use to people maintaining and extending Enterprise SOA. This would allow for the service metadata ontology to encompass additional service concerns.

Another area of interest is to develop an autonomous agent to gather metadata information from the organizational SOA to be represented in a graphical format. This would allow architects to quickly identify critical services, their dependencies and relationships with other services.

Acknowledgements

The authors wish to acknowledge King Fahd University of Petroleum and Minerals (KFUPM) for utilizing the various facilities in carrying out this research.

References:

[1] Ivar Jorstad, Schahram Dustdar, and Do Van Thanah, A Service Oriented Architecture Framework for Collaborative Services, *Proceedings of the 14th IEEE International Workshop on Enabling Technologies (WETICE' 05)*, IEEE 2005.

[2] Firdaus Bhathena, Combat Increasing IT Complexity, *Enterprise Architect Magazine*, Winter 2006.

[3] Peltz and C.Web, Services orchestration and choreography, *Computer*, Vol. 36, No. 10, Oct. 2003, pp. 46 – 52.

[4] Sayed Hashmi, Service-Oriented Architecture Explained, *O'Reilly*, August 2003, http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html

[5] Stan Kleijnen and Srikanth Raju, An Open Web Services Architecture, *Queue*, Vol. 1, No. 1, 2003, pp. 38-46.

[6] A. Shaikh Ali, O.F. Rana, R. Al-Ali, and D.W. Walker, UDDIe: An extended registry for Web services, *Proceedings of the Applications and the Internet Workshops*, 27-31 Jan. 2003 pp.85 – 89.

[7] Keita Fujii and Tatsuya Suda, Semantic – Based Dynamic Service Composition, *IEEE Journal on selected areas in communications*, Vol 23, No. 12, Dec 2005.

[8] Object Management Group's CORBA website http://www.omg.org/corba

[9] Microsoft's DCOM website http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndcom/html/msdn_dcomarch.asp

[10] Booth et, al. (editors) W3C Working Group Note 11: Web Services Architecture, World Wide Consortium (W3C), February 2004, http://www.w3.org/TR/ws-arch/#stakeholder

[11] Duane Nickull et. al., Service-Oriented Architecture Whitepaper Adobe Systems Incorporated 2005.

[12] Jim Barry, Jean Paoli, C. M. Sperberg-McQueen, Eva Maler, and Francois Yergeau, *Extensible Markup Language (XML) 1.0*, Third Edition, W3C Recommendation 04, Feb 2004 http://www.w3.org/TR/REC-xml

[13] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, *Web Services Description Language (WSDL)* 1.1; W3C Note 15, Mar 2001 http://www.w3.org/TR/wsdl

[14] Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers, UDDI *Spec Technical Committee Version 3*, Published 19, Oct 2004

[15] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Canon Henrik, and Frystyk Nielsen, *SOAP Version 1.2 Part 1: Messaging Framework*, W3C Recommendation 24 June 2003 http://www.w3.org/TR/soap12-part1

[16] Boris Lublinsky, SOA Design: Meeting in the Middle, *JavaPro Magazine 20*, Aug 2004 http://www.ftponline.com/javapro/2004_10/magazine/features/blublinsky

[17] Jiamao Liu, Ning Gu, Yuwei Zong, Zhigang Ding, and Quan Zhang, Service Registration and Discovery in a Domain-Oriented UDDI Registry, *The Fifth International Conference on Computer and Information Technology*, CIT 21-23 Sept. 2005, pp. 276 – 283.

[18] V. Tosic and H. Lutfiyya, Web Service Offerings Language (WSOL) Support for Context Management of Mobile/Embedded XML Web Services, *Advanced International Conference on Internet and Web Applications and Services*, 19-25 Feb. 2006, pp.156 – 156.

[19] David Beckett, Brain McBride, *RDF/XML Syntax Specification (Revised)*, W3C Recommendation 10 Feb. 2004, http://www.w3.org/TR/rdf-syntax-grammar

[20] DAML web site http://www.daml.org

[21] DAML+OIL Ian Horrocks et. al.; March 2001, http://www.daml.org/2000/12/daml+oil-index