www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 6 Issue 2 Feb. 2017, Page No. 20383-20387
Index Copernicus Value (2015): 58.10, DOI: 10.18535/ijecs/v6i2.39

An Approach for Composing Web Services through OWL

Kumudavalli.N
Mtech Software Engineering
kumudavalli.n@gmail.com

ABSTRACT

The semantic web promises to bring automation to the areas of web service discovery, composition and invocation. It purports to take the Web to unexplored efficiencies and provide a flexible approach for promoting all types of activities in tomorrow's Web. In this paper, we had proposed an ontology-based framework for composition of Web services. The model is also based on an iterative and incremental scheme meant to better capture requirements in accordance with service consumers' needs. OWL-S markup vocabularies and associated inference mechanism are used and extended as a means to bring semantics to service requests. This framework is used for exploring interesting Compositions of existing Web services. In this approach we look for similarities between Web services and this method is followed if we are unaware of specific goal for services. The framework first screens web services for composition leads based on their service operations.

Index Terms: Semantic Web, Web services, Service recognition, Service composition, Ontology, OWL

1. INTRODUCTION

The current trend in software architecture is to build platform-independent software Components, called Web services that are available in the distributed environment of the Internet. The Web is currently going through a transformation from a data-centric Web to a Semantic Web consisting of both self-describable data and Web services, which are a new type of first class object. The Web service deployment of previously isolated applications allows such an application to be described and published by one organization (i.e., service provider), and discovered and invoked later by other independently developed applications (i.e., service Consumers) [1], essentially making these applications interoperable on the Web. Nowadays, an increasing amount of companies and organizations only implement their core business and outsource other application services over Internet. Thus, the ability to efficiently and effectively select and integrate inter-organizational and Heterogeneous services on the Web at runtime is an important step towards the development of the Web service applications. This un-precedent ease of application integration contributed to the increasing popularity of Web service composition, which aims at providing value-added services through composing existing services. A key characteristics distinguishing this from traditional Web service composition approaches as governed by standards such as WSFLWSFL, XLANG, BPEL4WS, DAML-S and OWL-S is that it is driven by the desire to find any unanticipated and Interesting compositions of existing Web services. Traditional compositions approaches are usually goal driven that contain a fixed set of criteria. It then uses these criteria to search for matching component Web services. Since the goal provided by the user already implies what type of compositions the user anticipates, the evaluation of interested composition is not a major concern in these approaches. If we don't know specific goal then we need to address how to determine interesting and suitable service compositions. The simplest approach following this strategy would be an exhaustive search for compensability between all Web services. In this approach we look for similarities between Web services.

Basic concepts of web services are Web Services Definition language (WSDL) is an XML-based language, which specifies a Web service by defining messages that provide an abstract definition of the data being transmitted and operations that a Web service provides to transmit the messages. Four types of communication are defined involving a service's operation (endpoint): the endpoint receives a message (one-way), sends a message the endpoint receives a message and sends a correlated message (request-response), and it sends a message and

receives a correlated message (solicit-response).Operations are grouped into port types, which describe abstract end points of a Web service such as a logical address under which an operation can be invoked. A WSDL message element defines the data elements of an operation. XML Schema syntax is used to define platform independent data types which messages can use. Each message can consist of one or more parts. The parts can be compared to the parameters of a function call in a traditional programming language. The semantics of Web services is crucial to enabling automatic service composition. It is

Important to insure that selected services for composition offer the "right" features. Such features may be syntactic (e.g., number of parameters included in a message sent or received by a service).They may also be semantic (e.g., the business functionality offered by a service operation or the domain of interest of the service).To help capture Web services' semantic features; we use the concept of ontology. An ontology is a shared conceptualization based on the semantic proximity of terms in a specific domain of interest[3].Ontology are increasingly seen as key to enabling semantics-driven data access and processing .The y are expected

to play a central role in the Semantic Web, extending syntactic service interoperability to semantic interoperability. Issues to be considered while composing web services are:Composability model for Web services: A major issue in the composition of Web services is whether those services are composable [5]. Composability refers to the process of checking if Web services to be composed can actually interact with each other. We propose a composability model for comparing syntactic and semantic features of Web services.

Automatic generation of composite services is a technique to generate composite service descriptions while preserving the aforementioned composability rules. The proposed technique uses as input a high-level specification of the desired composition. This specification contains the list of operations to be performed through composition without referring to any component service.WSDL (Web Services Description Language) .WSDL is being standardized within the W3C consortium. Major industry leaders are supporting and participating in WSDL development. Hence WSDL will likely gain considerable momentum as the language for Web service description. However, WSDL provides little or no support for semantic description ofthat describe Web services from a syntactic point of view. To cater to Semantic Web-enabled Web services, we extend WSDL with semantic capabilities. This would lay the groundwork for the automatic selection and composition of Web services.

DAML+OIL adopts an object-oriented approach, describing ontology in terms of classes, properties and axioms.DAML+OIL builds on earlier Web ontology standards such as RDF and RDF Schema and extends those languages with richer modeling primitives (e.g., cardinality).Other Web ontology languages such as OWL [3] may also be used to specify the Web services. It mainly includes constructs proposed ontology. We model the proposed ontology using a directed graph. Nodes represent the ontology's concepts. Unfilled nodes refer to WSDL concepts (e.g., name, binding, input). Gray nodes refer to extended features introduced to augment WSDL descriptions with semantic capabilities Edges represent relationships between the ontology's concepts. They are labeled with the cardinality of the corresponding relationship. For example, the edge service → operation states that a service has one or more operations. The edge operation→input states that an operation has at most one input message. A Web service is defined by instantiating each ontology concept. We consider three types of participants in our approach: providers, composers, and consumers. Providers are the entities (e.g., credit reporting agency) that offer simple Web services (e.g., Credit History service).The provider is responsible for describing its Web service by assigning a value to each ontology concept services. Once generated, composite service descriptions are advertised in a service registry so that they can be discovered.

2.1 Web Service Ontology

We rely on OWL-S to define our Web services with WSDL grounding. We refer to
the applicability contained in the OWL-S service profile as locale in this paper.

To recognize the fact that certain services (e.g., payment) may be involved in multiple OWL-S categories of services (e.g., travel, Healthcare, legal), we use the concept of domain to group relevant operations, or more appropriately, operation interfaces.

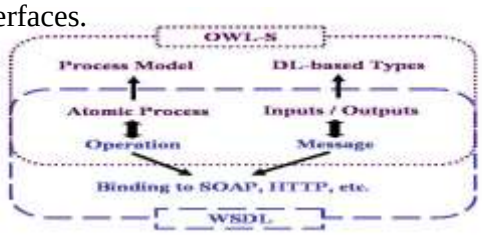

Fig. 1 GROUNDING OF OWL WITH WSDL

An operation interface specifies a shared functionality implemented by operations from different Web services. A Web service's involvement with a domain is reflected by whether it supplies or consumes an implementation of an operation interface in such a domain. We assemble a hierarchy of indices (FIG 1) to existing domain anthologies to unambiguously categorize the type of operation inputs and outputs. Requirements for ontology languages Ontology languages allow users to write explicit, formal conceptualizations of domains models. The main requirements are:

1. A well-defined syntax
2. A well-defined semantics
3. Efficient reasoning support
4. Sufficient expressive power
5. Convenience of expression.

The importance of a well-defined syntax is clear and known from the area of languages it is a necessary condition for machine-processing of information. All the languages we have presented so far have a well-defined syntax. DAML+OIL and OWL build upon RDF and RDFS and have the same kind of syntax. In this paper we are selecting OWL for semantic description of service.

2.1.1 The OWL Language

All this as lead to a set of requirements that may seem incompatible: efficient reasoning support and convenience of expression for a language as powerful as a combination of RDF Schema with a full logic. Indeed, these requirements have prompted W3C's Web Ontology Working Group to define OWL as three different sublanguages, each of which is geared towards fulfilling different aspects of these incompatible full set of requirement 1) OWL Full: The entire language is called OWL Full, and uses all the OWL languages primitives (which we will discuss later in this chapter). It also allows combining these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) to change the meaning of the pre-defined (RDF or OWL) primitives, by applying the language primitives to each other. For example, in OWL Full we could impose a cardinality constraint on the class of all classes, essentially limiting the number of classes that can be described in any ontology.

2) OWL DL: In order to regain computational efficiency, OWL DL (short for: Description Logic) is a sublanguage of OWL Full which restricts the way in which the constructors from OWL and RDF can be used The disadvantage is that we lose full compatibility with RDF: RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Conversely, every legal OWL DL document is still a legal RDF document.

3) OWL Lite: An ever further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointness statements and arbitrary cardinality (among others).The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is of course a restricted expressivity.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more-expressive constructs provided by OWL DL and OWL Full. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modeling facilities of RDF Schema When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations will be impossible. OWL syntax is based on XML it consists of Header, class elements, property elements, and property restrictions etc which are used for analysing and composing web services.

Sample owl schema is:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf -schema#"
xmlns:owl
="http://www.w3.org/2010/07/owl#"
xmlns="http://www.mydomain.org/african"
>
<owl:Ontology rdf:about="">
<owl:VersionInfo>
My example version 1.2, 17 October 2010
</owl:VersionInfo>
</owl:Ontology> <owl:Class
rdf:ID="animal">
<rdfs:comment>Animals form a class</rdfs:comment>
</owl:Class>
<owl:Class rdf:ID="plant">
```

<rdfs:comment>

Plants form a class disjoint from animals </rdfs:comment>
<owl:disjointWith="#animal"/> </owl:Class>

<owl:Class rdf:ID="tree"> <rdfs:comment>Trees are a type of plants</rdfs:comment>

<rdfs:subClassOf rdf:resource="#plant"/> </owl:Class>

Here class represent classes and class of represents sub classes and operation represents operations of services.

3 Recognition and Composition

Much like molecules in the natural world where they can recognize each other and form bonds in between [2], Web services and operations can also recognize each other through both syntax and semantics. Consequently, they can compose and bring about potentially interesting behaviours.

We identify two types of recognition are Operation recognition: direct and indirect recognition

Service recognition: promotion, inhibition. **Direct Recognition:** A direct recognition is established between operations opa and opb, if opa consumes an operation interface opintf , which is implemented by opb. In addition, opa and opb must be mode, binding and message compostable [5].

Indirect Recognition: A target operation opts indirectly recognizes a source operation ops, if ops generate some or all input parameters of opt. There is a potential need to relay parts of the output message from ops to parts of the input message to opt at the composition level. A bond is established between ops and opts for each input parameter opt can receive from ops. We denote the set of bonds between ops and opt as B(ops → opt). If we refer to the set of all operations that opt recognizes as OPs(→ opt), then Promotion When operation op1 of service sa produces an entity (i.e., output parameter)

5 Conclusions:

In this paper, we proposed a frame work to discover interesting compositions of existing Web services. This automatically screen for Web service compositions. We also presented the concept of interestingness of these compositions and proposed objective measures to evaluate it. This is useful to beginners who wanted to which particular services come under their requests.

that in turn provides service sb, we say that sa : op1 promotes sb. Inhibition Similarly, when operation op1 of service sa consumes

an entity (i.e., input parameter) that in turn provides service sb, we say that sa : op1 inhibits sb

• Exact match: na = nb
• Is-a: na is a child of nb
•Has-a: na has a component nb

We assume that the above relationships among parameter types are already declared in domain anthologies and thus can be automatically detected. Various measures [3] have been proposed to determine whether two operations are compostable at both syntactic and semantic levels. These measures can be used to determine whether a direct recognition-based composition is actually valid. For promotion and inhibition based compositions, they are valid because the entities of interest provide the corresponding services by declaration. In this section, we focus on how the validity of an indirect recognition based composition can be determined in the verification phase. We denote comp(OPs, opt) as an operation composition involving a set of source operations Ops providing input parameters to target operation opt, where

OPs ⊂ OPs(→ opt). In order for comp(OPs, opt) to be valid. Base on these concepts we identify similarity between objects.

4 Evaluations:

Not all service compositions discovered during the earlier phase are necessarily interesting and useful. The purpose of post screening analysis and evaluation is to identify those that are truly interesting and useful. For this we identify operation similarities and propose new service compositions from existing this is done by giving weights to the services. By parsing OWL schema we do all these things.

6 REFERENCES

[1] Web Services Architecture—W3C Working Group Note, http:// www.w3.org/TR/2004/NOTE-ws-arch-20040211/, Feb. 2004.

[2] Web Service Modeling Ontology, http://www.wsmo.org/, 2009.

[3]OWL-S: Semantic Markup for Web Services—W3C Member Submission, http://www.w3.org/Submission/OWL-S/, Nov.2004

[4] B. Medjahed, A. Bouguettaya, and A.K. Elmagarmid, "Composing Web Services on the

Semantic Web," VLDB J., Sept. 2003.

[5]. T. Andrews et al. Business Process Execution Language for Web Services (BPEL4WS) 1.1. Online: http://www-106.ibm.com/developer works/webservices/library/ws-bpel,May 2003.

[6]. W. Abramowicz, K. Haniewicz, M. Kaczmarek and D. Zyskowski "Architecture for Web services Filtering and Clustering", Internet and Web Applications and Services, (ICIW '07),May 13-19, 2007, Le Morne, Mauratius.