2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies

Effective Web Service Discovery Based on Functional Semantics

Demian Antony D'Mello [1] and V. S. Ananthanarayana [2]

[1] *Department of Computer Science & Engineering, St. Joseph Engineering College, Mangalore, INDIA – 575 028*

[2] *Department of Information Technology, National Institute of Technology Karnataka, Mangalore, INDIA – 575 025*

[1] demian.antony@gmail.com [2] anvs@nitk.ac.in

***Abstract*—Web service discovery is a mechanism which facilitates an access to the Web service descriptions. UDDI facilitates the discovery based on the service functionality through keyword and category matching. Such discovery techniques do not consider the semantics and user context as they are too syntactic in nature. In this paper, we propose a well formed functional semantics to describe an operation of a Web service. We design the extendible functional knowledge to map the requested or published operation descriptions into an abstract operation. The experimentation shows that, the proposed functional semantics based discovery mechanism has better performance in terms of precision and recall.**

I. INTRODUCTION

A Web service is an interface, which describes a collection of operations that are network accessible through standardized XML messaging [1]. Web service discovery is the mechanism which facilitates the requester, to gain an access to Web service descriptions that satisfy his functional requirements. UDDI [2] is the early initiative towards discovery, which facilitates both keyword and category based matching. The main drawback of such mechanism is that, it is too syntactic and does not represent user's context and semantics. In literature, many researchers have proposed different mechanisms in order to search suitable Web services based on the functional and non-functional behavior of Web services. In this paper, we present the Web service discovery mechanism, which explores more suitable Web services based on well formed functional semantics of the Web service operations.

The functional semantics based discovery mechanism described in [3] provides a unified way to semantically describe the functionality of Web service for the service providers and customers. The methodology described in the paper lacks effective (precise) discovery of Web services. The major limitations are:

1. The functional semantic description uses only single domain object instance. Thus the functional semantics will not represent the usage context of both the request and published operation in all scenarios. For example, consider travel scenario. The operation with an action "reserve" and an object "bus" is interpreted in two ways (two usage contexts): (a) reserve the bus for one day excursion. (b) Reserve bus for the travel from place x to y.
2. The functional semantics should support the use of qualifiers for objects as the qualifiers provide more information about the participating object of the service. For example, the operation descriptions like "get prime number' includes the qualifier *prime*.
3. The functional description of an operation sometimes contains the nouns (action nouns) with no explicit action associated with the domain object(s).

In this paper, we find the solutions for the key issues with respect to functional semantics based Web service discovery. The key contributions of this paper are:

1. The design of domain independent, extendible functional knowledge for the discovery process.
2. A well formed semantic rules to describe the functionality of Web service operations.
3. Matching mechanism for the Web service discovery based on the functional semantics.

The rest of the paper is organized as follows: In the next section, we present the definitions involved in describing functional semantics of Web service operations. Section 3 describes the functional knowledge structure to store the abstract operation descriptions and Web service publishing. In section 4, we discuss the experiment results. Section 6 draws the conclusions.

II. FUNCTIONAL SEMANTICS FOR WEB SERVICE OPERATIONS

A Web service is a network accessible software interface having collection of operations that aim at providing some kind of value to the consumers of the Web service. Thus Web service operation is nothing but the execution of appropriate action on specific object to provide value to the requester. The following definitions help to frame the functional semantics to describe the Web service operations.

A. Functional Semantics Terminology

Definition 1. *Generic Action* is an action used to perform the operation on an object or to get some kind of service in terms of object. For example in travel scenario, "perform bus booking" description involves the generic action "perform" which is commonly used across multiple domains.

Definition 2. *Specific Action* is an action performed on an object in a given domain to render service to the requester. For example in travel scenario, the description "reserve train ticket" involves the specific action "reserve" which is specific to travel, entertainment and accommodation domains.

Definition 3. *Domain Object* is an object of specific domain for which the required action is sought by the Web service operation. For example bus, hotel, room, ticket, taxi, flight etc are the objects found in travel and tourism domains.

We classify domain objects as *major objects* and *sub-objects* based on the association among them. The major objects are the entities (objects) that constitute the service domain. For example taxi, hotel, train etc are a few major

978-0-7695-3915-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ACT.2009.11

objects of the travel domain. The sub-objects are a part of specific major object in some domain of interest. For example ticket, room, seat etc. are a few sub-objects found in travel domain as "seat" and "ticket" is a part of major object "train".

Definition 4. *Domain Noun* is a noun which is specific to the service domain. For example reservation, availability, vacancy etc. are the domain nouns found in travel domain.

We classify domain nouns as *action noun* and *simple noun* based on the action represented by the domain noun. An action noun is a domain noun which has related specific action. A simple noun doesn't represent implicitly any specific or generic action. For example in travel scenario, the noun "reservation" is an action noun whereas the domain noun "availability" is simple noun.

Definition 5. *Qualifier* is a word which adds the value to the object or to the action noun. For example operation description "find air distance" has a qualifier "air". Similarly, "get prime number" contains a qualifier "prime".

Definition 6. *Abstract Operation* is an operation description which is an equivalent form of multiple operation descriptions.

All operation descriptions are transformed into their corresponding abstract operation(s) during Web service advertisement for the effective discovery. The important property of abstract operation description is that, it does not allow generic action together with an action noun to be present in its description.

B. Description of Abstract Web service Operations

The functional semantics approach uses the natural way of expressing the functionality of Web service operations i.e. operations are described in terms of actions, objects and nouns. Both the provider and requester of Web service use restricted natural form to express the Web service functionality i.e. operation description. Thus functionality of an operation can be described in the following *three* formats.

(i) OP_{Desc} = {(Generic Action)(Qualifier)*(Domain Object)$^{+}$(Domain Noun)}

(ii) OP_{Desc} = {(Specific Action)(Qualifier)*(Domain Object)$^{+}$}

(iii) OP_{Desc} = {(Qualifier)* (Domain Object)$^{+}$ Action Noun}

Consider the travel scenario; the following operation descriptions follow the rules of functional semantics.

(a) reserve train ticket (b) bus ticket booking

(c) check train availability (d) do flight ticket reservation

All operation descriptions are preprocessed before being mapped into abstract operations. The following rules guide the preprocessing of operation descriptions.

Rule 1: If the action noun is present along with the generic action then the generic action is replaced by the specific action which is related to the action noun and the action noun is eliminated from the description.

Rule 2: If the action noun is found in the operation description with no generic or specific action then the specific action of the action noun is considered as action eliminating the action noun.

As an illustration, consider the operation description as "do train ticket booking". The description contains generic action and action noun. The generic action is now replaced by "reserve" which is related specific action for action noun and the action noun is removed from the description. This results in a preprocessed operation description "book train ticket" which is considered as an abstract operation.

III. FUNCTIONAL KNOWLEDGE STRUCTURE AND ABSTARCT OPERATION MAPPING

To perform Web service discovery based on the functional description of Web service operations, we design an extendible functional knowledge which contains interdependent knowledge structures to represent the complete functional knowledge for all categories of Web services. The interdependent knowledge structures are: *Object List*, *Action List*, *Qualifier List* and *Noun List*.

1) Object List: Object list is a sorted list with finite elements where each element contains *four* fields i.e. information items. They are- *object name*, *object identifier*, *object type* and *a pointer* to the sorted related object list having similar/related names of a specific object. The object name refers to domain object for which action is to be sought, object identifier is a unique identification string and object type refers to either major (M) or sub-object (S). The object list and related object list can be implemented as dynamic array which are sorted based on the object name.

2) Action List: Action list is a sorted list with finite elements each containing *three* fields namely *action name*, *action identifier* and *a pointer* to the sorted related action list containing similar action words for a specific action. The action list and related action list can be implemented as dynamic arrays.

3) Qualifier List: Qualifier list is a sorted list with finite elements each containing *three* fields namely *qualifier name*, *qualifier identifier* and *a pointer* to the sorted related qualifier list containing similar qualifier words for a specific qualifier. The qualifier list and related qualifier list can be implemented as dynamic arrays.

4) Noun List: Noun list is a sorted list with finite elements each containing *five* fields namely *noun name*, *noun identifier*, *noun type*, *a pointer* to its corresponding action (if any) and *a pointer* to the sorted related noun list containing similar noun words used to describe specific noun. The noun list and related noun list can be implemented as dynamic arrays sorted based on the noun name. The noun type refers to two noun categories i.e. action noun (A) and simple noun (S).

Figure 1 depicts the partial functional knowledge structure showing interdependent structures having knowledge about different domains. The functional knowledge is augmented by the providers in order to improve the hit rate of their advertised services during the discovery to improve business.

To transform Web service operation description to its abstract operation equivalent, a separate list is maintained called *Abstract Operation List (AOL)*. The structure of AOL is defined below.

5) Abstract Operation List: The abstract operation list is a sorted dynamic array with finite elements each representing a abstract operation. The element contains *operation identifier,*

operation pattern and *Web service count;* where operation pattern is a string of finite length which contains fixed length identifiers of objects, nouns, qualifiers and actions. The Web service count refers to the number of Web services having an operation which maps to the abstract operation. The operation pattern is generated for each abstract operation defined in AOL. Let M be the fixed length for identifiers of actions, nouns, qualifiers and objects. The first M characters represent the action identifier; next, the sets of M characters represent the qualifier identifiers (optional), finally the sets of M characters of operation pattern represent the domain object identifiers followed by the noun identifier (optional).

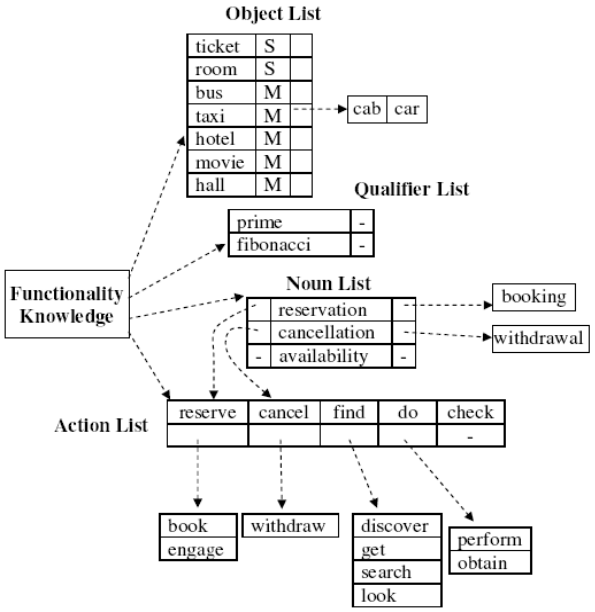

Figure 1. Functional Knowledge Structure for Discovery

A. Web Service Description based on Functional Semantics

Let Profile(WS) be the profile of the Web service to be published into UDDI registry. Profile(WS)={service-desc, binding-desc} where, service-desc refers to service specific descriptions like service name, business name, operation descriptions etc and binding-desc refers to binding details like URL for the access. service-desc = {service-name, business-name, OP_{List}} where, OP_{List} is the list of operations and their descriptions supported by the Web service. OP_{List} = {opr_1, opr_2…opr_N} where, opr_i is the description of an operation. opr_i = {opr-name, desc-list, info-list} where desc-list is the functional semantics of operation description in the defined format and info-list is optional additional information to update the extendible functional knowledge. desc-list={action, qualifier(s), object(s) noun} where qualifiers and noun are optional. info-list = {action-set, qualifier-set, object-set, noun-set} where, action-set contains similar action words, qualifier-set contains similar qualifier names for a given qualifier and object-set contains similar object names and noun-set contain similar noun names.

IV. EXPERIMENTS AND RESULTS

We have conducted several experiments as a proof for the functional semantics based matchmaking concept. We use a collection of 19 Web services having total of 30 distinct operations from XMethods [4] service portal (http://www.xmethods.com) and divided them into THREE categories (Email and IP Address related services, Zip code, Phone and Address related services, Weather services). We frame 33 service discovery requests based on their short natural language descriptions from Web Service Definition Language (WSDL) files (operation names). We compare the performances of proposed mechanism with FunWSDS system [3]. The recall of the proposed mechanism is high as compared to FUNWSDS system. Both the mechanisms exhibit 100% precision if the published and requested operations are described with incorrect functional semantics. Figure 2 show the average recall values for the experiments conducted for three different service categories.

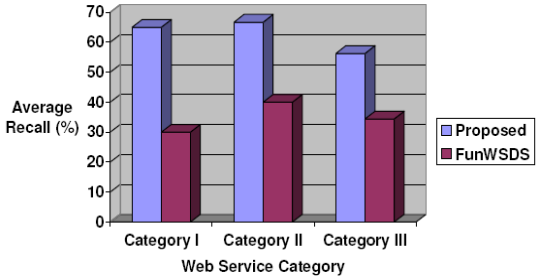

Figure 8. Performance Evaluation (Average Recall)

V. CONCLUSION

Web service discovery is an important activity which explores functionally similar services for the given service discovery request. In this paper, the authors propose a well defined functional semantics to describe the Web service operations for the publishing and discovery. The paper designs an extendible functional knowledge structure for the effective Web service discovery. Several experiments are conducted for the Web service descriptions listed in XMethods portal. The experimentation reveals that, the use of functional semantics in Web service operation description will improve the effectiveness (Recall and Precision) of Web service discovery.

REFERENCES

[1] H. Kreger, "Web Services Conceptual Architecture (WSCA 1.0)", Published May 2001, [online] Available: www.ibm.com/software/solutions/webservices/pdf/wsca.pdf, [visit: April 2007].

[2] Riegen, C.V. (Ed), 2002. *UDDI Version 2.03 Data Structure Reference* [online]. OASIS Open 2002-2003. Available from: http://uddi.org/pubs/DataStructure-V2.03-Published-20020719.htm [Accessed 8 November 2007].

[3] Lei Ye, Bin Zhang, "Discovering Web Services Based on Functional Semantics", In Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06), IEEE 2006.

[4] XMethods, [online] Available: http:\\www.xmethods.com, [visit: February 2009].