A Tree Structure for Efficient Web Service Discovery

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@gmail.com [2] anvs@nitk.ac.in

***Abstract*—Web service discovery is a mechanism to obtain descriptions of Web services based on the requester's functional or non-functional requirements. UDDI is the first step towards meeting the requester's functional requirements. The Web service architecture involving UDDI does not provide an efficient and effective discovery mechanism. The UDDI is large repository and its information is either distributed or replicated at several sites to improve fault tolerance. To improve the performance of UDDI based Web service discovery, we propose the broker based architecture for Web service publishing and discovery. The broker of the architecture stores the abstract Web service information and the functional knowledge for an effective Web service discovery. The paper explores Service Operation Tree (SOT) to store Web service information in a compact way which also speeds up the discovery process. The authors also analyse and illustrate the discovery mechanism backed by empirical results.**

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 satisfies his functional or non-functional requirements. UDDI [1] [2] is the early initiative towards discovery which facilitates both keyword and category based matching. The Web service architecture involving UDDI does not provide efficient and effective discovery as the UDDI is solely responsible for publishing and discovery mechanism. The UDDI is large repository and its information is either distributed or replicated at several sites to improve fault tolerance. Such distributed architecture of UDDI suffers from performance issues like low response time and throughput. The discovery process implemented on such architecture loads the entire page [3] consisting either partial or whole Web service description into main memory for the matchmaking process. UDDI based Web service discovery is too syntactic and does not represent the semantics of functionality supported by various published 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 Web service operations. We propose broker based architecture for Web service publishing and discovery. The broker of the architecture stores the abstract information of all published Web services in a compact way. 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 the next subsection we give brief overview of different architectures and discovery mechanism proposed in literature.

A. *Evolution of Web service Architectures*

The architectures proposed in literature can be classified based on the location where the Web service information (distributed or centralized) is stored. We classify the Web service discovery architectures broadly as *Centralized Architecture* and *Distributed (de-centralized) Architecture*.

In centralized Web service discovery architectures [4] [5], the Web service descriptions are stored in a central repository (registry). For static discovery, the Web service descriptions can be accessed through Web portals such as WebServiceList (www.webservicelist.com), XMethods.net and Remote Methods (www.remotemethods.com). The Web service request is made to service registry which is responsible for the request processing and discovery mechanism. The major problem with such architecture is that, the central registry becomes bottleneck of processing which suffers from stress problems such as performance degradation and scalability.

In distributed architectures (de-centralized architecture), the Web service descriptions are normally stored at the provider's site and the service request is executed by the server or by the coordinating agents by gathering Web service information which is available at the several sites. The distributed (decentralized) architecture can be further classified based on the nature of information access methods and the Web service information distribution structure. We classify the distributed architectures as: (a) *Internet (Web) based architectures* (b) *Agent based architectures* (c) *P-2-P architectures*.

In Internet based architectures [6] [7], the search engine (Web crawler) techniques are used to discover the Web services. The WSDL crawler [7] is designed to discover the Web services directly from the Internet. The agent based architectures [8] [9] use software agents for intelligent and quality driven Web service discovery. The agents assist the discovery mechanism by processing the requests and finding Web services satisfying the desired quality [8]. The agents are also used for communication and accessing Ontology towards smart discovery of Web services [9]. The decentralized approaches based on Peer to Peer networks have been proposed by many researchers to solve the problems of centralized approach. The P-2-P discovery mechanism [10] [11] [12] is a kind of discovery technique emerging based on the peer-to-peer network construction. Peer-to-peer systems provide a scalable alternative to centralized systems by distributing the data and load among all peers. These systems are decentralized, scalable and self-organizing. The decentralization of WS discovery will increase the fault tolerance and search efficiency. Figure 1 shows the taxonomy of Web service discovery structures.

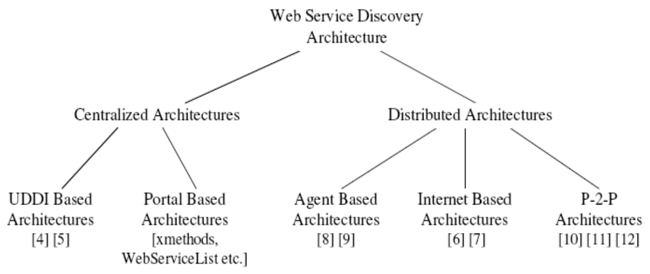
Web Service Discovery Architecture
Centralized Architectures
Distributed Architectures
UDDI Based Architectures [4] [5]
Portal Based Architectures [xmethods, WebServiceList etc.]
Agent Based Architectures [8] [9]
Internet Based Architectures [6] [7]
P-2-P Architectures [10] [11] [12]

Figure 1. Taxonomy of Web service Discovery Architectures

B. Evolution of Web service Discovery Techniques

In literature, different methods are proposed to capture syntactical, semantical and contextual similarity of discovery request and service advertisements for the discovery. We classify the Web service discovery methods into *two* broad categories as *functional property* based and *non-functional property* based approaches. In functional property based Web service discovery, the functional i.e. behavioral details of Web services are used for matchmaking of discovery request with the published i.e. advertised profiles of Web services. The non-functional criterions like QoS [13], trust [14], usability [15] and personalization [16] are also used for effective Web service discovery. We classify functional property based discovery mechanisms based on the nature of information retrieved or stored for the matchmaking. They are *syntax based matching*, *behavior based matching* and *semantic matchmaking* methods.

The syntax based Web service discovery mechanisms normally find the matching information for the request from the WSDL documents of advertised services. The matchmaking is commonly based on keyword/string, service category, interface and input/output [1] [17] [18] [10] of the published Web services. The behavior based matching mechanisms [19] [20] discover Web services based on the internal process structure of the service or service requester's behavior. The semantic based matchmaking mechanisms find Web services based on the service semantics. The matchmaking is mainly performed based on the semantic descriptions like information theoretic information [21], functional semantics [22], Ontological concepts [23] context information [24] and goal (Input, Output, Pre-condition, Effect) [25] of advertised Web services.

C. Motivation and Contribution

The major problems associated with UDDI based centralized Web service discovery mechanism are:

a) UDDI suffers from performance degradation(i.e. response time) as, all the Web service advertisements have to be loaded into main memory for the matchmaking which is time and space consuming process.
b) UDDI employs keyword/string and category based matchmaking process which is too syntactic in nature and the stored UDDI information does not represent semantics of service or operation functionality.

In this paper, we address these issues and the contribution of this paper is:

a) Design of service knowledge (Tree structure) to represent Web services of repository
b) A broker based architecture for effective and efficient Web service discovery
c) Effective and efficient Web service discovery mechanism to find functionally similar Web services.

The rest of the paper is organized as follows: In the next section, we present functional semantics model to describe Web service operations. Section 3 describes the service knowledge structure to store the abstract Web service information. Section 4 presents the broker based architecture for Web service discovery. Section 5 describes the Web service publishing and discovery mechanism. In section 6, we discuss the experimentation and results. Section 7 draws the conclusions.

II. Modeling of Web Service Knowledge

A Web service can advertise multiple operations. The widespread adoption of Web services has resulted in Web services having same or similar functionality by providing some common operations. To store the Web service operations in an efficient way for the discovery, we define *two* data structures called Web service list (WSL) and Service Operation tree (SOT).

1. Web Service List (WSL). A Web service list is a sorted dynamic array having *four* fields namely, Web service key (*ws-key*), *ws-id* (unique identifier generated by the broker), *ws-link* and *sot-link* where ws-id is a Web service identifier, ws-link is a pointer to the Web service entry in WSL having same set of operations and sot-link is the pointer to the leaf node of Service Operation Table (SOT) which corresponds to a link to its operations in the SOT or a pointer to the predecessor Web service in WSL having same operations.

2. Service Operation Tree (SOT). A service operation tree is a binary tree with each node consisting *five* fields. They are operation identifier (*opr-id*) which specifies the operation identifier of Web service operation (abstract operation); child pointer (*child-link*) which is a pointer to the remaining operations of a Web service; sibling pointer (*sibling-link*) is a pointer to the list operations which shares a common operation prefix; parent pointer (*parent-link*) is a pointer to its predecessor node; Web service link (*wsl-link*) is a pointer to the WSL entry to which opr-id is the last operation in the sorted advertised operation list. The root node of SOT is labeled with T and has only child-link which points to Web various service operation sequences.

The property of SOT is that at any node X of SOT, the opr-id at X will not be repeated at the child or sibling branch which is linked to X.

As an illustration consider *four* Web services having N (N>0) operations to be advertised into service repository. Let

Opr_1 to Opr_{10} be the operation identifiers of abstract operations obtained after mapping them into virtual operations. Consider five web services with abstract operations as follows. WS_1 = {Opr_1, Opr_2, Opr_3, Opr_4, Opr_5, Opr_6, Opr_7}, WS_2 = {Opr_1, Opr_2, Opr_4, Opr_8, Opr_9}, WS_3 = {Opr_1, Opr_2, Opr_4, Opr_8, Opr_9, Opr_{10}}, WS_4 = {Opr_2, Opr_4, Opr_6, Opr_7, Opr_8, Opr_9, Opr_{10}} and WS_5 = {Opr_2, Opr_4, Opr_6, Opr_7, Opr_8, Opr_9, Opr_{10}}. Figure 3 shows the SOT and WSL after insertion of service operations of Web services. The numbers within circles (node) indicate the operation identifiers (Opr_1 to Opr_{10}) in an ascending order. In WSL, WS_1 to WS_5 represent Web service key followed by unique WS-Id numbers for Web services.

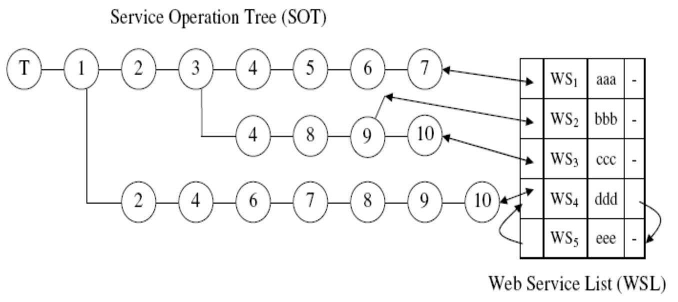

Figure 3. Service Operation Tree and WSL for Published Web Services

The insertion of new Web service having N operations is formally presented in Figure 4 (Algorithm Insert-SOT-Operation).

III. THE BROKER BASED ARCHITECTURE FOR DISCOVERY

We propose broker based architecture for Web services for the effective and efficient Web service discovery. Figure 5 depicts different roles and operations supported by the broker based architecture. The architecture assumes that the Web service requesters and providers use functional semantic rules to describe advertised and requested Web service operations.

We define an additional role to the conceptual Web service architecture [1] named *broker* and a new operation namely *quick find (search)* and *register*. The broker is defined between Web service registry and requester/provider which facilitates both the requester and provider to specify the need and advertisements in terns of functional semantics. From the architectural perspective, the broker is a middleware which can be implemented as a Web service. The quick find operation is defined between the broker and requester, which effectively discovers the Web services from SOT. The register operation is defined between the provider and broker for the functional semantics based Web service publishing.

We define the broker with *four* internal components namely *Service Publisher*, *Service Discovery Engine, Functional Knowledge* and *Service Knowledge*. The service publisher component facilitates the registration, updating and deletion of business and Web service related information. The main functionality of service discovery engine is to discover Web services satisfying requester's functional demands. The functional knowledge is interlinked data structure which represents extendible knowledge (refer section II.B) updated by various Web service providers. This component facilitates the effective Web service publishing and discovery. The service knowledge represents SOT and WSL of published Web services (refer section III). The service knowledge is an abstract representation of All published Web services which facilitates the efficient (response time/ throughput) discovery.

```
Algorithm Insert-SOT-Operation
Input: List of Operations (Adv_Opr_List) of a Web service
       Number of Advertised Operations (N), Web service key (WS-Key)
Output: Updated SOT & WSL and ws-id (Web service Identifier)
Begin
   Let Adv_Opr_List = {$Opr_1$, $Opr_2$…$Opr_N$} be the sequence of
   operation identifiers such that, $Opr_1 < Opr_2 < … < Opr_N$)
   Let T be the root of the SOT
   Generate ws-id and insert ws-key and ws-id into WSL
   If (T has single node) then
      Create a child branch of T involving all $Opr_i$ of Adv_Opr_List
      Store the corresponding WSL entry into wsl-link of the last node
      Store the address of the last node into sot-link of WSL
   Else
      Search for operation sub-sequence starting from T (prefix) of Adv_Opr_List
      If (no sub-sequence is found) then
         Let $T_C$ be the child of T
         Create a skewed tree by linking operation nodes through child-link
         Attach the skewed tree as a new sibling branch of $T_C$
         Store the corresponding WSL entry into wsl-link of the last node
         Store the address of the last node into sot-link of WSL
      Else
         Let $T_D$ be the last matching operation node
         If (the wsl-link of $T_D \neq$ NULL) then
            Find the WSL element linking the last node
            Scan and establish the link between predecessor and successor -
                WSL entries through ws-link till NULL is reached
            Replace the NULL value with the address of newly inserted ws-id
         End If
         Create a skewed tree by linking the remaining operation nodes through child-link
         If (child-link of $T_D$ = NULL) then
           Link the skewed tree to $T_D$ as a child-link branch
         Else
           Let $T_E$ be the child node (at child-link) of $T_D$
           Link the skewed tree to $T_E$ as a sibling-link branch
         End If
         Store the corresponding WSL entry into wsl-link of the last node
         Store the address of the last node into the corresponding sot-link of WSL
      End If
   End If
   Retuen ws-id
End
```

Figure 4. Inserting Service Operations of a New Web Service

IV. WEB SERVICE PUBLISHING AND DISCOVERY MECHANISM

In this section, the authors present the Web service publishing and discovery mechanism.

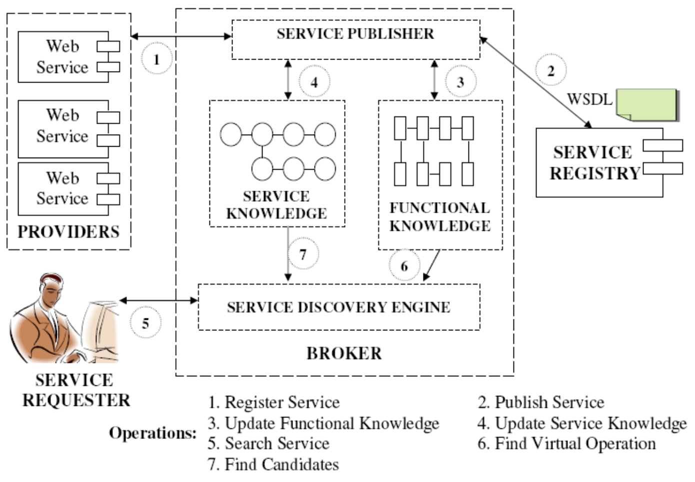

Figure 5. The Broker based Web service Architecture

A. *Extension of WSDL 2.0 Document*

We extend the WSDL 2.0 structure to publish the Web services with functional semantics as follows. We select the *documentation* element of the WSDL to insert the information which is necessary for the effective service discovery. We define a new tag called *operationDesc* to insert the functional semantics of all abstract operations present in the Web service. The functional semantics for the “Fibonacci number generation service” is depicted in Figure 6.

```
<?xml version="1.0" encoding="utf-8" ?>
<description>
<documentation>
<operationDesc xmlns:xf ="http://www.w3.org/2001/XMLSchema-Instance"
           xf:schemaLocation="http://www.nitk.ac.in/itdept/descSchema/desc.xsd"
           xmlns="http://www.nitk.ac.in/itdept/descSchema">
    <operationList>
        <operation>
            <operationName>Generate-Fibbonaci-Number</operationName>
            <operationType>C</operationType>
            <semantics>
                <action>generate </action>
                <qualifier>fibonacci</qualifier>
                <object>number</object>
            </semantics>
        </operation>
    </operationList>
    <information>
        <action name="generate">
            <related>obtain</related>
            <related>compute</related>
        </action>
    </information>
</operationDesc>
</documentation>
<types>…</types>
<interface>…</interface>
<binding url="www.numbertheory.com">…</binding>
<service name="FibonacciNumber"></service>
</description>
```

Figure 6. Extended WSDL Document of a Web Service

B. *Web Service Publishing*

The provider of the Web service publishes the extended WSDL 2.0 into the UDDI through the broker. The steps involved in the Web service publishing are presented below.

1. The broker processes the WSDL to obtain the service name, binding details, the abstract operation details like operation name and functional semantics.
2. The broker middleware publishes the Web service into UDDI registry and obtains the service key.
3. The abstract operation descriptions are preprocessed according to Rule 1 & 2 as defined in section II.
4. For each preprocessed abstract operation in WSDL, the operation pattern is generated by obtaining the appropriate action, object, qualifier and noun identifiers from the functional knowledge. If the action/noun/qualifier and object is not present in the functional knowledge, then these are inserted by generating appropriate identifiers. Now the operation pattern is generated concatenating the identifiers as defined in section II.
5. Search the operation pattern in VOL. If found, return the operation identifier of the operation having the generated operation pattern otherwise insert the operation as a new virtual operation along with operation pattern into VOL.
6. All the operation identifiers of all Web service abstract operations along with service key are inserted into SOT and WSL as described in the algorithm (Figure 4).

C. *Web Service Discovery*

The Web service discovery for the service discovery request and the matchmaking process is summarized below.

1. The service request is preprocessed according to Rule 1 & 2 (section II) to retrieve the functional requirement.
2. The action list, qualifier list (if required), object list and noun list (if required) of the functional knowledge are searched to get the corresponding identifiers. Unavailability of any identifier results in discovery failure.
3. On locating required identifiers from the functional knowledge, the operation pattern for the discovery request is formed. After building the operation pattern, the pattern is searched in VOL. If the pattern is found then the corresponding operation identifier is retrieved from the VOL otherwise discovery failure is reported.
4. The abstract Web service information of all published Web services is searched by traversing SOT for the requested operation identifier (refer Figure 7) and all the Web services having requested operation (operation identifier) are returned to the requester as the functionally suitable Web services.

D. *Analysis of Discovery Mechanism*

The discovery process is time critical activity. Here we present the asymptotic analysis of the proposed Web service discovery mechanism.

Let W be the action/object/qualifier/noun words present in the respective list of functional knowledge and R be the related words for the action/object/qualifier/noun. The worse case word comparisons for any list are: $W \log_2 W \log_2 R$. Let K be the number of objects present in lengthy request. Now the worse case word comparisons are: $(K+3)\, W \log_2 W \log_2 R$. Thus worse case complexity is: $\Theta\, (K\, W \log_2 W \log_2 R)$.

Let A be total number of virtual operations present in Virtual Operation List (VOL). The worse case comparisons of operation sequence of requested operation with VOL are $\log_2 A$. Thus, the complexity is: $\Theta\, (\log_2 A)$.

Let M be the number of nodes (operations of published Web services) present in Service Operation Tree (SOT). The worse case comparisons to discover all possible candidates are M. Thus complexity of SOT searching operation is Θ (M). Thus complexity discovery mechanism is = $\Theta\, (\mathrm{Max}\, \{K\, W \log_2 W \log_2 R,\ \log_2 A,\ M\})$. For $W > R > M > A > K$, the complexity is $O\, (KW\log_2 W)$ i.e. $O\, (W\log_2 W)$.

V. IMPLEMENTATION AND EXPERIMENTS

The proposed broker based Web service discovery mechanism is implemented on the Windows XP platform using Microsoft Visual Studio .NET development environment and Microsoft visual C# as a programming language. The broker is designed and implemented as a standalone visual program which interacts with the provider and requester through different interface forms. The service repository is implemented as a Web service which in turn communicates with the SQL server 2000 database. The database table is created to store the information about the published Web services.

```
Algorithm WebServiceDiscovery
Input: A list of requested service operations
Output: Web service keys or failure report
Initialize Cand-List [] = NULL // Global list
Begin
    For each requested operation, obtain the operation identifier (op-id) from VOL
    If op-id is not found then
        Report "Discovery Failure" and Exit
    Else
        Sort the operations based on op-id
        Let Opr$_1$, Opr$_2$, ..Opr$_M$ be the sorted list of operations
        For each Opr$_i$ do
            Let T be the root of SOT, Node=child-link (T)
            Find-Candidates (SOT, Node, Opr$_i$)
            If (Cand-List=$\Phi$) then
                Report "Discovery Failure" and Exit
            End If
    End If
End

Procedure Find-Candidates (node, opr)
Begin
    If (Node = NULL) then
        Return
    End If
    If (op-id (Node) = opr) then
        Traverse_Inorder(node, opr);
    Else
        Find_Cand(child-link(node), opr);
        Find_Cand(sibling-link(node), opr);
    End If
End

Procedure Traverse-Inorder(node, opr)
Begin
   Traverse the child branch until ws-link =NULL or sibling-link!=NULL
   Let X be the such node
   If (ws-link (X) !=NULL) then
      Traverse the WSL through ws-link of X
      Record the candidate Web services from WSL
   End If
   If (sibling-link (X) !=NULL) then
      Traverse-Inorder(sibling-link(X), opr);
   Traverse-Inorder(X, opr);
End
```

Figure 7. Discovering Web services from SOT & WSL

We have conducted several experiments as a proof for the functional semantics based discovery mechanism. We have also performed preliminary experiments to evaluate the system performance in terms of precision and recall.

Precision = (Relevant ∩Retrieved) / Retrieved

Recall = (Relevant ∩ Retrieved) / Relevant

We use a collection of 14 Web services having a total of 26 distinct operations from XMethods [26] service portal (http://www.xmethods.com) and divided them into THREE categories. Table 1 shows the categories, Web services and their operations.

We frame 34 service discovery requests based on their short natural language descriptions from Web Service Definition Language (WSDL) files (operation names). We compare the performance of proposed mechanism with FunWSDS system [22]. Figure 8 show the average recall values for the experiments conducted for *three* different Web service categories.

TABLE 1. WEB SERVICE CATEGORIES AND THEIR OPERATIONS

Category (I, II, III)	**Web Services**	**Supporting Operations**	**Requests Created**
Information services (ISBN & Date)	3	6	11
Financial Services (Stock quote & currency change)	6	13	15
Holiday services	5	7	8

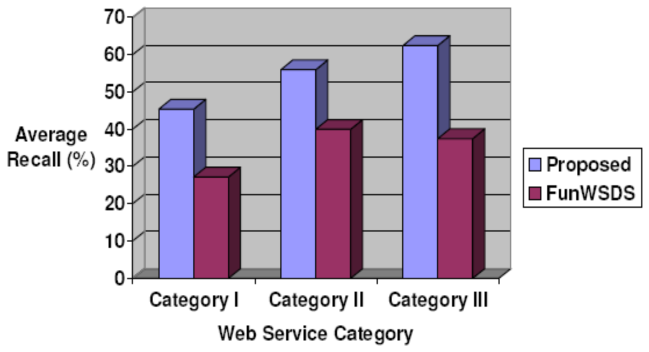

Figure 8. Average Recall and Precision Values

The Recall of the discovery mechanism is less than 100% because of following reasons.

(1) For some operations, it is difficult for the publisher to frame the description as per the functional semantics.
(2) Requester sometimes may not follow the functional semantics in a precise way.
(3) Certain scenarios (operations) can be described in multiple ways (forms).

The Recall of the proposed mechanism is high as compared to FunWSDS system since the proposed systems consider multiple objects and qualifiers to be present in the operation descriptions. Thus proposed method is effectively discovers the Web services for a given task.

We use same set of 14 of Web services as listed in XMethods having 41 operations and represent them using SOT. The SOT representation takes only 23 nodes which yield compression ratio of 21%. Since the abstract operation of each Web service is stored in the main memory of the broker, the discovery mechanism need not load the entire page having the information of advertised Web services. This will substantially reduce the secondary memory access time which in turn improves the response time of the discovery mechanism. This is because the SOT acts as an index to the Web service descriptions present in the registry.

The Web service publishing is not a time critical process. The necessary information required for the matchmaking is pre-computed by the broker during Web service publishing. The pre-computation involves the mapping of abstract operations into virtual operations and finding the operation pattern for the abstract operation from the VOL. Since the Web service discovery is a time critical event, the requested operation description is mapped into virtual operation through construction of operation pattern. The SOT structure is then traversed to obtain all the Web services offering (having operation identifier) requested operation. We see that, the discovery of Web services involves search operation on the functional and service knowledge which is always stored in main memory of broker.

VI. CONCLUSION

Web service discovery is an important and time critical activity which explores functionally similar services for the given service request. The paper proposes broker based architecture for Web service publishing and discovery. The necessary information required for the matchmaking is pre-computed during Web service publishing and stored at the broker for an efficient discovery of Web services. The proposed mechanism uses a well formed functional semantics to describe Web service operations in an effective way. This improves the Recall and Precision of discovery as compared to keyword/string matching based discovery. The broker based Web service discovery architecture is implemented and 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 and response time) of discovery as compared to UDDI based 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] V. S. Ananthanarayana and K. Vidyasankar, "Dynamic Primary Copy with Piggy-Backing Mechanism for Replicated UDDI Registry, Lecture Notes in Computer Science (LNCS), Vol. 4317/2006, 389-402, Springer Berlin / Heidelberg.

[4] Chen Wu, Elizabeth Chang and Ashley Aitken, "An empirical approach for semantic Web services discovery**",** In proceedings of the 19th Australian Conference on Software Engineering, pp. 412-421, IEEE 2008.

[5] Ziqiang Xu, Patrick Martin, Wendy Powley and Farhana Zulkernine, "Reputation-Enhanced QoS-based Web Services Discovery", In Proceedings of the 2007 IEEE International Conference on Web Services (ICWS 2007), IEEE 2007.

[6] Eyhab Al-Masri and Qusay H. Mahmoud, "Discovering Web Services in Search Engines", IEEE INTERNET COMPUTING, MAY/JUNE 2008.

[7] Chen Wu and Elizabeth Chang, "Searching services on the Web": A public Web services discovery approach", In Proceedings of the Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, pp. 321-328, IEEE 2008.

[8] Tiantian Zhang, Junzhou Luo and Weining Kong, "An Agent-based Web Service Searching Model", In Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design Proceedings, pp. 390-395, IEEE 2005.

[9] GUAN Hong-jie, MENG Fan-rong, SUN Jin-fei and DU Peijun, "Web Service Discovery Based on the Cooperation of UDDI and DF", In Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08, pp. 1-4, IEEE 2008.

[10] Fatih Emekci, Ozgur D. Sahin, Divyakant Agrawal and Amr El Abbadi, "A Peer-to-Peer Framework for Web Service Discovery with Ranking", In Proceedings of the IEEE International Conference on Web Services (ICWS'04), IEEE 2004.

[11] S. Sioutas a, E. Sakkopoulos, Ch. Makris, B. Vassiliadis, A. Tsakalidis, P. Triantafillou, "Dynamic Web Service discovery architecture based on a novel peer based overlay network", The Journal of Systems and Software 82 (2009) pp. 809–824.

[12] S. Sioutasa, E. Sakkopoulos, L. Drossos and S. Sirmakessis, "Balanced distributed web service lookup system", Journal of Network and Computer Applications 31 (2008) 149–162.

[13] Ran, S., 2003. A Model for Web Services Discovery with QoS, *SIGecom Exchange*, ACM, 4(1):1-10.

[14] Ali, S.A., Ludwig and Rana, O.F., 2005. A Cognitive Trust-based Approach for Web Service Discovery and Selection, *In Proceedings of the Third European Conference on Web Services (ECOWS'05)*, IEEE Computer Society 2005.

[15] Hyun Namgoong, Moonyoung Chung, Kyung-il Kim, HyeonSung Cho andYunku Chung, "Effective Semantic Web Services Discovery using Usability", In Proceedings of the ICACT 2006, pp. 2199-2203, IEEE 2006.

[16] Nomane Ould Ahmed M'Bareck and Samir Tata, "How to consider requester's preferences to enhance web service discovery?", In Proceedings of the Second International Conference on Internet and Web Applications and Services (ICIW'07)", IEEE 2007.

[17] Wang, Y. and Stroulia E., 2003. Flexible Interface Matching for Web-Service Discovery. *In Proceedings of the 4th International Conference on Web Information Systems Engineering (WISE'03)*, IEEE Computer Society.

[18] Bin Xu, Tao Li, Zhifeng Gu and Gang Wu, "SWSDS: Quick Web Service Discovery and Composition in SEWSIP",In Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06), IEEE 2006.

[19] Bensheng Yun, Junwei Yan and Min Liu, "Behavior-based Web Services Matchmaking", In Proceedings of the 2008 IFIP International Conference on Network and Parallel Computing, pp. 483-487, IEEE 2008.

[20] E. Stroulia and Y. Wang, "Structural and Semantic Matching for Accessing Web Service Similarity,"International Journal of Cooperation Information Systems", Vol. 14, pp. 407 - 437, 2005.

[21] Mourad Ouzzani and Athman Bouguettaya, "Efficient Access to Web Services", IEEE INTERNET COMPUTING, MARCH - APRIL 2004, pp.34-44, IEEE 2004.

[22] 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.

[23] Po Zhang and Juanzi Li, "Ontology Assisted Web Services Discovery", In Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), IEEE 2005.

[24] YE Qi, SONG Guo-Xing, "Context-Aware Service Discovery in Pervasive Computing Environments", In Proceedings of the Third International Conference on Semantics, Knowledge and Grid, pp. 556-557, IEEE 2007.

[25] Martin, D. et-al, 2005. Bringing Semantics to Web Services: The OWL-S Approach, *In Proceedings of the SWSWPC 2004*, Springer LNCS 3387, pp. 26-42.

[26] X Methods, [online] Available: http:\\www.xMethods.com, [visit: February 2009].