Extended Service Registry to Support I/O Parameter-Based Service Search

H.N. Lakshmi and Hrushikesha Mohanty

Abstract The increasing availability of Web services within an organization and on the Web demands for an efficient search mechanism to find services satisfying user requirements. Universal Description, Discovery, and Integration (UDDI) is an industry standard for Service Registries, developed to solve the Web service search problem. However, UDDI offers limited search functionalities which may return a huge number of irrelevant services. Also, often consumers may be unaware of precise keywords to retrieve the required services satisfactorily and may be looking for services capable of providing certain outputs. In this paper, we propose a new system called ESR for extended and efficient service search using an Object Relational Database. The experimental results demonstrate the efficiency of service search in our extended service registry (ESR) and the variety of user queries supported.

Keywords Service Registries · Service search · UDDI · I/O parameters

1 Introduction

Web services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. As growing number of services are being available, selecting the most relevant Web service fulfilling the requirements of a user query is indeed challenging. Various approaches can be used for service search, such as searching in Universal Description, Discovery, and

H.N. Lakshmi (✉)
University of Hyderabad, Hyderabad, India
e-mail: hnlakshmi@gmail.com

H.N. Lakshmi · H. Mohanty
CVR College of Engineering, Hyderabad, India
e-mail: hmcs_hcu@yahoo.com

Integration (UDDI), Web and service portals. In this paper, we take up the issue of searching at Service Registries for its practicality in business world as providers would like to post their services centrally, as searching there is less time-consuming than searching on World Wide Web.

Basically, current technology supports service search by name, location, business, bindings, or TModels and binds two services based on composability of their protocols. The search API is limited by the kind of information that is available and searchable in UDDI entries and do not provide any support for complex searches like I/O parameter-based search and automatic composition of Web services. Also, our experiments show that average response time of current UDDI implementations increase with a substantial increase in number of services registered.

The above shortcomings of UDDI have motivated us to build an extended service registry (ESR) system capable of offering powerful and efficient search operations. We propose the use of Object Relational Database as repository of Web services. Information about the Web services, extracted from their WSDLs, is stored in tables and relational algebraic operators are used for service search. This work is an extension of our previous work [1], where we proposed a RDBMS approach for Service Registries.

Often consumers may be unaware of exact service names that are fixed by service providers. Rather consumers being well aware of their requirements would like to search a service based on their commitments (inputs) and expectations (outputs). Based on this concept, we have explored the feasibility of I/O-based Web service search in our proposed ESR system, to support varying requirements of the consumer. Utility of such an I/O-based Web service search for composition of Web services is shown in our previous work [1].

The rest of the paper is organized as follows. In Sect. 2, we describe the Object Relational Schema used for service registry. This is an extension of our previous work [1]. Section 3 discusses our experimental results. In Sect. 4, we essay the related work. We conclude our work in Sect. 5.

2 Object Relational Database for Service Registry

In this section, we first give an overview of how Web services are registered in UDDI. We then argue for the need of Object Relational Database for storing Web service information in a registry. We then propose an Object Relational Schema for our ESR.

2.1 WSDL and UDDI Relationship

The Web Services Description Language (WSDL) is a XML language for describing Web services. It contains service interface defined as a set of operations and messages, their protocol bindings, and the deployment details. UDDI (V3) offers the industry a specification that is used for building flexible, interoperable XML Web services registries useful in private as well as public deployments [2]. If offers clients and implementers a comprehensive and complete blueprint of description and discovery foundation for a diverse set of Web services architectures.

There are four primary data types in a UDDI registry—businessEntity: used to represent the business or service provider within UDDI, businessService: used to represent business descriptions for a Web service, bindingTemplate: contains the technical information associated to a particular service, and tModel : used to define the technical specification for a Web service. A businessEntity can contain one or more businessServices. A businessService can have several bindingTemplates and each bindingTemplate contains a reference to one or more tModels.

WSDL document contains six major elements that defines Web Services—types: provides data-type definitions, message: provides an abstract definition the data being and consists of logical parts, portType: defines a set of abstract operations each referring to an input message and output message, binding: specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType, port: specifies an address for a binding and service: aggregates a set of related ports.

A summary of the mapping is as follows:

1. WSDL portType element is mapped to UDDI tModel.
2. WSDL binding element is mapped to UDDI tModel.
3. WSDL port element is mapped to UDDI bindingTemplate.
4. WSDL service element is mapped to UDDI businessService.

2.2 Object Relational Database as Service Registry

We observe that there are many sub-elements in WSDL that are not mapped to UDDI types such input message, output message, and message parts. Although these details can be added in tModel of the service, there are no supporting search functionalities provided in UDDI standards. This leads to the limited and poor search functionalities supported by UDDI. We, hence, propose to include these details in an ESR, enabling the registry to support additional search functionalities.

Each operation in a service has an input and output message, each of which in turn may have one or more message parts. For example, a HotelBooking service may take {Period, City} as input and provide {HotelName, HotelCost} as output. Thus, we require a strategy to store these multi-valued inputs and output messages in the registry. Such a structure clearly outs the principles of normalization.

Since current UDDI implementations store the UDDI data types in a Relational Database, a first thought would be to include the message details in a Relational database. A normalized Relational database solution to this requires that we store input and output message parts of each operation across multiple rows. With such a design, it takes multiple SQL Join operations to access and display the input or output message parts of a single operation. This implies reduced database performance. To avoid the need for multiple joins and to speed up the query, we propose to use multi-valued attributes for storing input and output message parts in our database design and, hence, use an Object Relational Database for our proposed ESR. Advantage of this approach is that it supports querying for services efficiently and also supports complex queries involving input and output parameters.

2.3 Extended Service Registry

In this section, we shall describe the Object Relational Schema for storing Web services in the registry. The relationship between the various tables in the proposed schema is depicted in the ER diagram in Fig. 1. This is an extension of our previous work [1].

1. Each Web service is given a unique ID (WSID) and stored with its name (WSName), port address (WSPortAdd), and operation name (OPName) in a Web service table (WSTable).

   WSTable: { WSID, WSName, WSPortAdd, OPName}

2. The parameters' table (ParTable) contains all the parameters, which take part as either input or output in any of the Web services in the registry, with their names in (PName). Each parameter is given a unique ID, (PID).

   PartTable : {PID, PName}

3. The Web service input/output table (WSInOutTable) lists all Web services in the registry with their respective input parameters (InPars) and output parameters (OutPars). InPars and OutPars are of collection type (ParList) and are stored as nested tables.

   WSInOutTable: {WSID, InPars, OutPars}

4. The schema also includes a query table for storing the user query (QueryT) and a table to store services matching the user query (MWSTable).

   - QueryT: {PNo, OutPars} where OutPars are the PIDs of output parameters specified in user query.
   - MWSTable: {WSIDs}

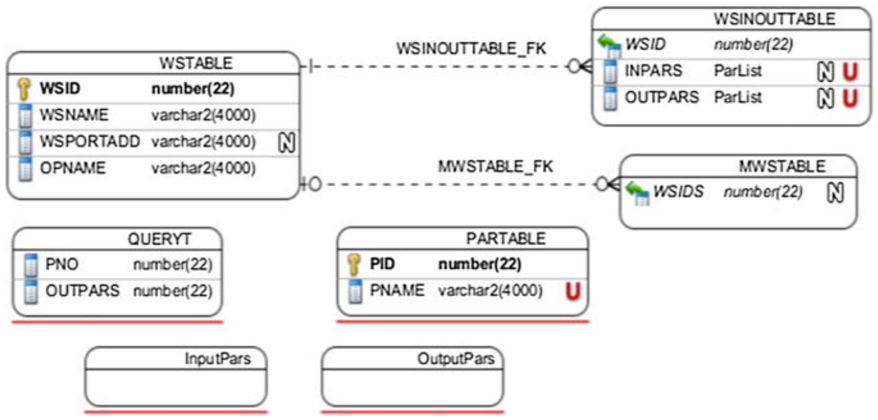

Fig. 1 ER diagram for service repository

```
Algorithm 1: Output Parameter-based Service Search
Input: $Q^O$, WSInOutTable: table
Output: MWSTable: table
foreach ParName in $Q^O$ do
    Select PID form ParTable where PName = ParName
    INSERT PID into the QueryT table
foreach ws in WSInOutTable do
    if $ws^O = Q^O$ then
        INSERT ws as Exact Match in MWSTable
    else if $ws^O \subset Q^O$ then
        INSERT ws as Partial Match in MWSTable
    else if $ws^O \supset Q^O$ then
        INSERT ws as Super Match in MWSTable
    else
        continue
```

2.4 *I/O Parameter-Based Service Search*

A Web service, *ws*, has typically two sets of parameters—set of inputs ws^I and set of outputs ws^O. When a user searches for a service providing a requested set of outputs and/or accepting a requested set of inputs, there may be many matching services in the registry. To categorize these matching services, we have defined various degrees of matching for services, based on Input/Output Parameter match, in our previous work [1] as follows:

1. Exact Match: ws_i is an Exact match of ws_j if the input/output parameters of ws_i exactly matches all the input/output parameters of ws_j, i.e.,

$$ws_i^O = ws_j^O \text{ or } ws_i^I = ws_j^O,\ i \neq j.$$

2. Partial Match: ws_i is a Partial match of ws_j if the input/output parameters of ws_i partially matches the input/output parameters of ws_j, i.e.,

$$ws_i^O \subset ws_j^O \text{ or } ws_i^I \subset ws_j^I,\ i \neq j.$$

3. Super Match: ws_i is a Super match of ws_j if the input/output parameters of ws_i is a superset of the input/output parameters of ws_j, i.e.,

$$ws_i^O \supset ws_j^O \text{ or } ws_i^I \supset ws_j^I,\ i \neq j.$$

2.5 Process of Parameter-Based Service Search

To empower the Service Registries with additional search capabilities, we define algorithms for I/O parameter-based service search. Algorithm 1 presents pseudocode for output parameter-based service search. Q^O represents output parameters specified in user query. Similar procedure is used for input parameter-based service search.

3 Experimental Results

The effectiveness of the OR Databases approach is shown by conducting two sets of experiments:

1. Performance of basic keyword search in Service Registries—jUDDI and ESR approach.
2. Scalability of Service Registries.

3.1 Experimental Setup

To evaluate the response time of UDDI-based service directories, we query a service directory that is implemented in jUDDI. Apache jUDDI (pronounced "Judy") is an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI v3) specification for Web services [3]. We conducted experiments on WSC Dataset [4]. We varied the number of Web services that were stored in Service Registries implemented in jUDDI and ESR, from 100 to 500 and then to 1,000. Five queries were submitted to the service directory in both the sets of experiments, and the response times of the service directory were computed. We ran our experiments on a 1.3-GHz Intel machine with 4-GB memory-running Microsoft Windows 7. Our algorithms were implemented using Oracle 10g and JDK 1.6. Each query was run 5 times and the results obtained were averaged, to make the experimental results more sound and reliable.

3.2 Comparison of Basic Keyword Search in Registries—JUDDI Versus ESR Approach

Web service search is performed in UDDI-based service directories using predefined APIs. find service() function in inquiry API, of jUDDI, is used to locate specific services in service registry and returns a serviceList structure that matches the conditions specified in the arguments. The various arguments supported are authInfo, businessKey, findQualifiers, and name. The default behavior of UDDI with respect to matching is exact match.

In the first set of experiments, we published 500 Web services in jUDDI and ESR. We then queried jUDDI with five different user requirements. Experiments were done for both approximateMatch and exactMatch Qualifiers. Same set of requirements were then fed to ESR and their response times were recorded. The experiment was repeated by publishing 1,000 Web services in both the registries. Figures 2 and 3 shows the performance results for Queries Q1 to Q5 on both the Service Registries for approximateMatch and exactMatch Qualifiers respectively.

In the second set of experiments, we varied the number of services published from 100 to 1,000 in jUDDI and ESR. We then queried jUDDI with four different user requirements. Experiments were done for both approximateMatch and exactMatch Qualifiers. The same sets of requirements were then fed to ESR, and their response times were recorded. Figure 4 depicts how the execution time for Query Q1 varies on both the Service Registries, with an increase in the number of services registered.

From the results obtained, we can infer that the time taken for search with approximate match is far lesser in our ESR approach when compared to the time taken in jUDDI. Though the time taken for service search with exact match appears to be close to the time taken in jUDDI, it is so only for the cases when

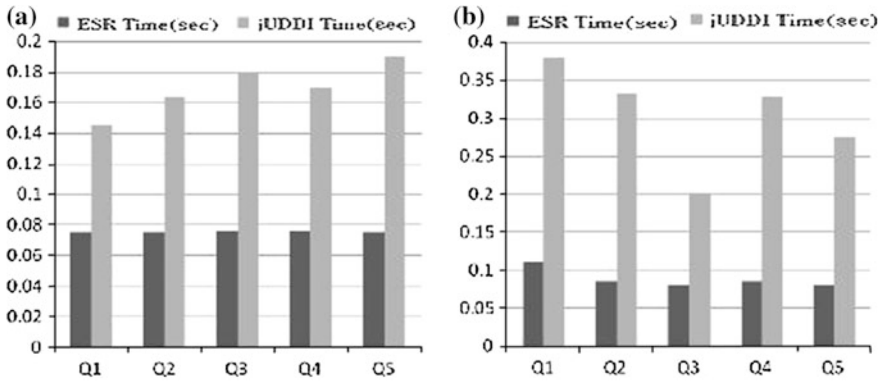

Fig. 2 jUDDI versus ESR for 500 services. **a** Exact match. **b** Approximate match

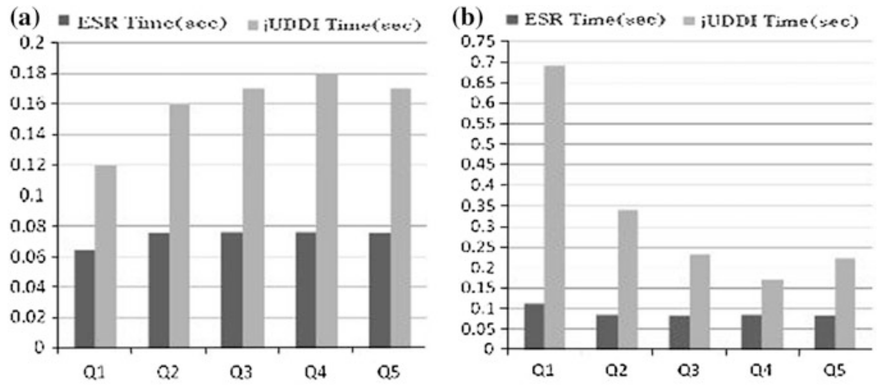

Fig. 3 jUDDI versus ESR for 1,000 services. **a** Exact match. **b** Approximate match

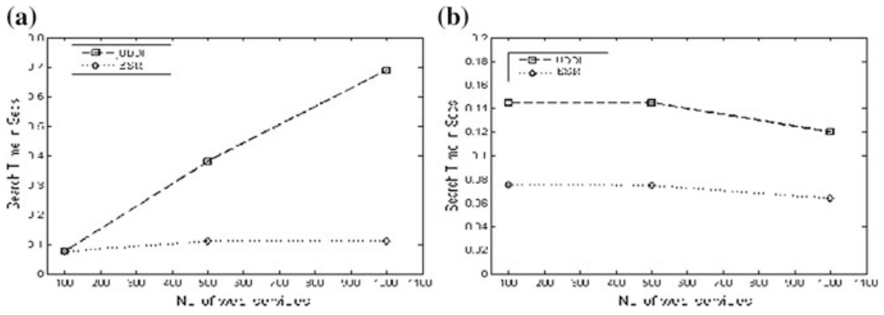

Fig. 4 Service search time in jUDDI versus ESR. **a** Exact match. **b** Approximate match

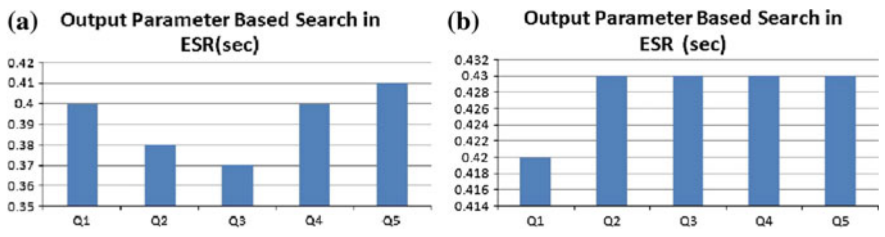

Fig. 5 Output parameter-based search in ESR. **a** Registry with 500 services. **b** Registry with 1,000 services

search results have very few matching services, meaning, for search results that have many matching services, our approach works faster than jUDDI, both in the case of approximate match and exact match. Also, the search time becomes more efficient in our approach with a substantial increase in number of services in the registry. Thus, we can finally conclude that service search in ESR is more scalable and efficient than searching in UDDI.

3.3 Performance of Output Parameter-Based Service Search

To empower the ESR with additional search capabilities, we defined algorithms for I/O parameter-based service search as discussed in Sect. 2.5. To include I/O parameter-based service search, we implemented Algorithm 1 and evaluated its performance using different output parameter patterns as user queries. In the first set of experiments, we published 500 Web services in ESR. The experiment was then repeated by publishing 1,000 Web services in ESR. Figure 5 shows the performance of output.

4 Related Work

In this section, we survey current e orts related to UDDI extensions and clustering Web services. Many efforts have been made to extend UDDI to improve service search, either by storing additional information about Web services typically by extending the WSDL format or by extending the API of UDDI registries to support additional functionalities.

Goodwin et al. [5] propose an architecture for semantic sensor matchmaker to make the service search and integration more efficient. To support addition of new query types, Mili et al. [6] proposed a generic extension framework to UDDI registries, by adding a middle tier acting as a broker between standard UDDI registries and clients. Juric et al. [7] incorporated version information into the businessService and tModel data structures. Zhou et al. [8] proposed UX (UDDI eXtension), a system in which the requesters QoS feedback is stored in a local database. Ran [9] proposed a new Web services discovery model with four roles: Web Service supplier, Web Service consumer, Web Service QoS certifier, and the new UDDI registry. Their model includes quality of service parameters along with functional parameters of Web services.

Most of the existing works propose to extend UDDI to incorporate different aspects related to semantics, or QoS. In this paper, we propose an Object Relational Schema for storing services in registry, to enable the registry to include multi-valued elements of WSDL and also to support I/O parameter-based service search.

5 Conclusion

Query-based Web service search is an important issue, especially for non-semantic Web services. Traditional UDDI-based service search lacks the ability to recognize all the features described in WSDLs. This leads to limited and poorly designed search operations that these registries offer. Experiments show that the performance of such service directories deteriorates with a substantial increase in number of services. Also, service search operations need to be extended to support varying requirements of the consumer. In order to improve the scalability and to empower Service Registries with additional search capabilities, we propose the use of Object Relational Databases as repository of Web services. We have simulated the algorithms on WSC Dataset [4]. The experimental results demonstrate the benefits of our ESR approach on varying user queries.

In the future work, we would like to work on a strategy for choosing the best matching service among the many candidate services. We further plan to integrate the current proposal with our previous work [1], to generate all possible compositions for a given user requirement, when expressed in terms of I/O parameters.

References

1. Lakshmi, H.N., Mohanty H.: RDBMS for service repository and composition. In: Fourth International Conference on Advanced Computing (ICoAC), 13–15 Dec 2012
2. UDDI Specifications: http://uddi.org/pubs/uddi_v3.html
3. Apache jUDDI:http://juddi.apache.org/index.html
4. The Web Service Challenge (WS-Challenge): http://www.ws-challenge.org/
5. Goodwin, J.C., Russomanno, D.J., Qualls, J.: Survey of semantic extensions to UDDI: implications for sensor services, SWWS, 16–22 (2007)
6. Mili, H., Tamrout, R.B., Obaid, A.: JRegistry: an extensible UDDI registry. Reports of NOTERE, pp. 115–128 (2005)
7. Juric, M.B., Sasa, A., Brumen, B., Rozman, I.: WSDL and UDDI extensions for version support in web services. J. Syst. Softw. **82**, 1326–1343 (2009)

8. Zhou, C., Chia, L., Lee, B.: QoS-aware and federated enhancement for UDDI. Int. J. Web Serv. Res. (IJWSR) **1**, 58–85 (2004)
9. Ran, S.: A model for web services discovery with QoS. ACM Sigecom Exchanges **4**(1), 1–10 (2003)