International Journal of Computer Applications (0975 – 8887)
Volume 3 – No.12, July 2010

An Efficient Web Service Discovery Architecture

Shrabani Mallick, D. S Kushwaha
Dept. of Computer Science & Engineering
Motilal Nehru National Institute of Technology
Allahabad

ABSTRACT

There is been a continuous effort by the software developers to reduce cost and time to deliver the software and this has led to the advancements in service oriented paradigm. Organizations across all spectra have already deployed their main operations to the Web, which has brought about a fast growth of various Web services. This has dramatically increased the need to build a fundamental infrastructure for efficient deployment and access of the exponentially growing repository of Web services. Probably the most important aspect in dynamic web service access is the web service discovery. The challenge remains in handling and processing request queries and presenting the requester the most appropriate set of web service interfaces. Many approaches and frameworks have been proposed to discover web services. Some of the approaches assume that the requests are placed in SOAP compatible formats whereas some works are based on handling the plain text queries by their semantics while others focus on key word based query processing. We have tried to formulate an approach that uses the Principle of Compositionality of language to derive interface descriptions and aggregate or filter for their appropriateness. Going by lexical and syntactic structure of a language, a skeletally parsed corpus of a written piece of text has been used to resolve the plain text query. This paper also tries to propose an architecture based on x-SOA that the organizes the method of web service discovery in an efficient and structured manner using a intermediary, requester friendly layer called the Request Analyzer(abbreviated as RA) between the service provider and service requester via a service broker. We describe how the RA facilitates the processing of a plain text request query to finally being resolved to a most appropriate web service. We propose an algorithm for a complete cycle of web-service discovery. A cache based service broker approach has been proposed that consumes even lesser time towards discovery path. A reputation based mechanism has also been incorporated for keeping track of the trustworthiness and type of license of a web service being used and hence ensuring the authenticity for future use in terms of their past performances.

Keywords - Web services, Principle of Compositionality, Request Analyzer (RA), Broker cache, x-SOA

1. Introduction

Service Oriented Architecture is an architectural paradigm that is used to build infrastructures enabling those with needs (consumers) and those with capabilities (providers) to interact via services across disparate domains of technology and ownership. Services act as the core facilitator of electronic data interchanges yet require additional mechanisms in order to function. Several new trends in the computer industry rely upon SOA as the enabling foundation. The basic SOA architecture involves handling service request, service discovery and service invocation. The success of service invocation depends largely on efficient analysis of the request query and discovering the most appropriate set of web service interfaces called as web service discovery.

Web service discovery is the process of locating, or discovering, one or more related documents that describe a particular XML web service using the Web Services Description Language (WSDL). It is through the discovery process that web service clients learn that a web service exists and where to find the XML web service's description document. Three key XML based standards have been defined to support web service deployment and use namely SOAP, WSDL and UDDI. SOAP defines a communication protocol for Web services. WSDL enables service providers to describe their applications. UDDI (Universal Description Discovery & Integration) offers a registry service that allows advertisement and discovery of Web services. The service consumer or web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. Whichever service the service-consumers need, they have to take it into the brokers, then bind it with respective service and then use it. They can access multiple services if the service provides multiple services. Many approaches and models have been proposed that describes different methodologies for efficient discovery of web services using OWL-S [5], language semantics[3]. Many of them actually don't talk about a framework for the discovery mechanism [1,3,4,6]. The process of service discovery is not only a step or state but is actually a process that involves many entities and states. It may not be necessary that service user or consumer is not SOAP/XML savvy in that case the discovery of a process becomes difficult.

There is a growing consensus and significant progress in both text understanding and spoken language understanding by investigating those phenomena that occur most centrally in naturally occurring unconstrained materials and by attempting to automatically extract information about language from very large corpora. Such corpora are beginning to serve as important research tools for investigators in natural language processing (NLP), speech recognition, and integrated spoken language systems, as well as in theoretical linguistics.

We have used a lightweight NLP approach by parsing the request query into constituent parts of speech wherein the verbs are used to derive the interface names and the set of identified interface names are analyzed against the nouns and adjectives for their appropriateness. The Request Anlayzer in between the Service Requester and Broker is being introduced that will offload the request pre-processing functionalities of the broker. The consumer can raise its request in plain text concealing the fact that in what form it will be published. The RA will act as an intermediary between the consumer and brokering agent.

2. Related Work

A lot of hype around web services and equating them to software as a service has motivated many works in this field. In

[1] it is proposed that Dynamic web Service invocations and hence selection are not independent operations but rather composite in nature and impose some form of ordering thus proposes that reliability is an issue for a composite web service invocation. It uses a FSM approach to draw the order among the operations in a given WS. The work mainly addresses the reliability issue of a composite web service selection, given a set of available web services.

In [2] a three layer Service brokering architecture for an underlying transparent web service access to the service client is presented. An Extended SOA (xSOA) architecture to support capabilities like service orchestration, “intelligent” routing, provisioning, and service management. A nice description of Metadata exchange for discovery of service description directly from the service is explored. An extension of this architecture is presented in our work.

In [3] semantics based request query analysis using a tree-form of data structure to discover the web services by assigning weight values to each node of the tree is discussed. Based on the weights assigned, the semantic similarity is computed between web service requested and web service registered. It cites a methodology for identifying the most similar web services but not the most appropriate ones. Moreover the quality of web services and the service level agreement remains an area of concern for the requester.

In [4] a keyword based approach is implemented triggered by the partitioning approach that is used in database design. The idea is used to cluster relevant and irrelevant services according to the user query which in turn helps the user to relieve itself from the burden of selecting web services from a huge set. The key approach is to cluster the services into a group of learned latent variables which is achieved by computing the probability. But plain keyword based search lack semantics and there is an overhead of high communication cost.

In [5] a web service discovery model, based on abstract and lightweight semantic web services descriptions, using the Service Profile ontology of OWL-S is focused. Goal is to determine an initial set of candidate web services for a specific request which can then be used for fine-grained discovery approaches. A web service matchmaking algorithm has been proposed extending object-based matching techniques allowing the retrieval of web services based on subsumption relationships and structural information of OWL ontologies the exploitation of web services classification in Profile taxonomies, performing domain-dependent discovery. No proper structured discovery framework is defined.

In [6] a neural network based approach is used that is best known for their ability for generalization. A novel neural network model is proposed that is used for classification of most suitable web services.

The classification strategy [9] is based on Quality of Web Services (QWS) parameters. The limitation in this work is that Suitability classification based only on the QWS parameters of web services might not suffice to users’ exact requirements.

In [7] an attempt to make web services available to a mobile client asynchronously that are resource poor has been proposed so that it need not wait while its request is being processed and returned. An asynchronous mode of call for the mobile users is beneficial so that synchronicity does not become a barrier to them in accessing web services. Since reaching a mobile agent through HTTP is problematic, SMS protocol is used for sending back the SOAP response. But the quality of service, trust and security aspect is not addressed. The discovery is again based on keyword match approach.

In [8] Multi-agent based approach on semantic web service discovery and ontology management. An approach for semantic web service discovery and propagation based on semantic web services and FIPA multi agents is proposed. The key idea is to expose the semantic interoperability of web service provider and agent. Based on the analysis of user request an ontology management is also proposed. Nothing has been specifically talked about the type and nature of user request. Moreover it requires the agent framework to be registered as an agent service in the FIPA Directory. No methodology has been proposed for tracking and keeping the trustworthiness of the Semantic web services.

Most of the work done by the previous researchers have tried to address the problem of web service discovery problem through approaches based on key words, semantics, neural networks, ordered service search, agent based framework etc. But who will be responsible for the preliminary processing of a service requests that happen before it is being handovered to the service broker remains a question. Many of them assume that the user is aware of the web service interfaces names, which is not always true. An organized framework which contains a layer exclusively for processing a simple plain text query based on their linguistic compositional semantics and getting back the most appropriate set of web services is what is focused here.

3. Proposed Work

Our proposed architectural framework is inspired by x-SOA for the web service discovery mechanism based on the linguistic compositional semantics of a request query. The main highlight of our framework is the intermediate layer the Request Analyzer (RA) which provides interoperability between the service requester and the service broker. We also propose an algorithm for the various stages involved in the process of service discovery.

The subsequent section discusses the architecture, algorithm and the interaction issues of the proposed work.

3.1. Architecture

The architectural framework is an enhancement over the existing 3-tier SOA architecture [2] consisting of: -

- Service Requester
- Request Analyzer
- Service Broker
- Service Provider

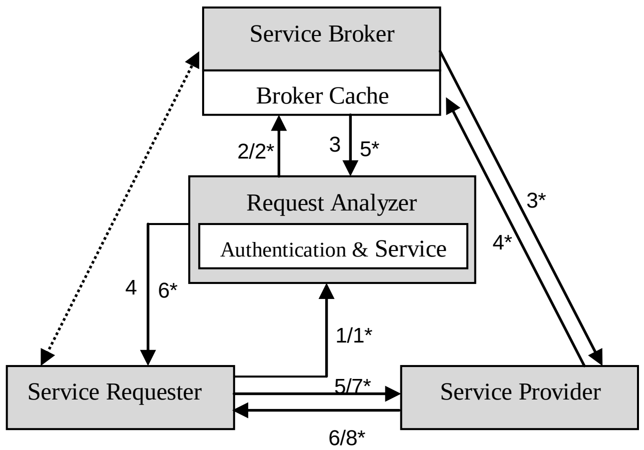

Figure 1: Proposed Architecture (LWSDM)

The new layer RA sits on top of a service requester that facilitates the preprocessing activities of a request query before it is being handed over to the broker for discovery. The roles and functionalities of the Request Analyzer (RA) can be enumerated as follows:

- The RA shall act as an intermediary between the service broker and the service requester.
- The RA will accept the request from the Requester and parse the request query string into the lexical components- verbs, noun, adjectives and adverbs.
- The RA will generate different data structures for the constituent verbs, nouns and adjectives+adverbs.
- It follows the **Principle of Compositionality** (that states that the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them)
- The RA will first forward the verb queue to the service broker which will in turn look up its cache or broadcast on the cloud whichever is applicable.
- On getting back the list of web service names (WSNs) from the brokering agent the RA will compare the WSNs with nouns and adjectives+adverbs that will serve as the parameters for context based and use-based matching.
- After comparison the most appropriate web service name will be returned back to the Requester.
- The Requester then invokes the interface.
- In addition, the RA will have a feedback loop from the Requester that will eventually be used to maintain an authentication database to know the trustworthiness of the web services being used and a service type info indicating their license agreement viz – prototype, free, pay per use or pay per period. A client might finally bind and acquire ownership of a web service.

3.2. Proposed Algorithm

The algorithm describes various phases for a complete cycle of a web service discovery which is interlaced on the proposed framework. A request query raised will serve as the input, it will be analyzed for its linguistic composition, the verbs will be looked up in the broker cache and if matching interface names are not found then they will be published on the web. The similarity and appropriateness of the matching service names will be calculated based on the nouns and adjectives and appropriate set of web services will be returned. Hence the overall algorithm can be broken down into 5 phases-

- Raise Request
- Analyze Request
- Lookup
- Publish
- Return Service

Algorithm: Layered Web Service Delivery Mechanism (LWSDM)

input: Request text

output: Appropriate set of web services

(i) Raise Request

```
        Let R be the request raised
        R={w} set of words
```

(ii) Analyze Request

```
        [Let V be the set of verbs, N be the set of Nouns, A
        be the set of adjectives & adverbs]
        count=0;
        for each word w in R
        $if\ w \in V\ then$
            $Q1 \leftarrow w;$ //store the verbs
        $elseif\ w \in N\ then$
            $Q2 \leftarrow w;$ //store the nouns
        $elseif\ w \in A\ then$
            $Q3 \leftarrow w;$ //store the adjectives & adverbs
        else discard;
```

(iii) Look up

The Broker cache has two look table

- NLT (The name Look Up Table)
  NLT{name_id} // containing the web service names (WSNs)
- ILT (The Interface Look Up Table)
  ILT{name_id, parameter_list} // A mapping of the WSNs to their parameters

```
    for(i=1; i<=count; i++)
      for each entry in NLT
        $if(name\_id = Q1[i])\ then$
         $return(list)$ //the matching WSN
        else
         $store\ Q1[i] in\ Brokerlist$
         $B[i] = Q1[i]$ //names not found in cacheNLT
        $call\ Broker\ (B[i])$//for Broadcast
```

(iv) Publish

```
        while B[i] has entries
            publish(B[i]);
                $list \leftarrow \{name\_id\}$
        return (list )  //the list of matching service names
        returned by the Provider
```

(v) Return Service

```
        for all name_id in the list
            calculate similarity based on the set of nouns Q2
            and set of adjectives Q3 and the parameter_list of
            ILT
return(service);  //the set of most appropriate WSN to the
Requester
```

The Broker cache shall have as indicated in the algorithm two look-up tables namely-

- NLT- The name Look-up table
- ILT-The Interface Look-up table

There will be a one-to-one mapping between the tables see Figure 2. The AST (Authentication and Service Type) database will be with the Request Analyzer that will maintain the reputation and service type information (viz. prototype, pay per use or pay per period) of the web services used. The requester may finally bind a service by acquiring ownership of it. The cache look-up tables will be updated periodically and AST shall be updated based on the feedback loops.

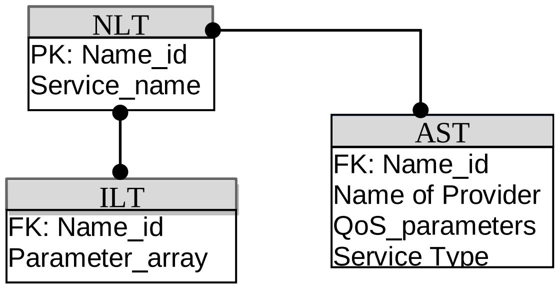

Figure 2: The Relationship diagram

3.3. Interaction between layers

The Request analyzer will be built as a middleware on a client machine. Figure 3 shows the interaction diagram between the various entities in the framework. In addition to the activities mentioned in the diagram, a reputation based feedback system shall be implemented by the Request Analyzer to calculate the trust values of the web services. The reputation information will be based on the past performance of a selected web service that can be used as an inherent aspect in determining its future behavior.

The selected web services can be weighted against their trust values before usage to reduce the overhead of rejecting a web service after using it. Common QoS metrics for determining the reputation may be ABA (abandonment rate) that is the percentage of calls abandoned while waiting to be answered, ASA (average speed to answer) means average time it takes for a call to b answered by the service desk, TSF (time service factor) implies percentage of calls answered within a definite timeframe, FCR(first call resolution) states percentage of incoming calls that can be resolved without the use of a callback or without having the caller callback to finish resolving the case and TAT (Turn around time) is the time taken to complete a certain task.

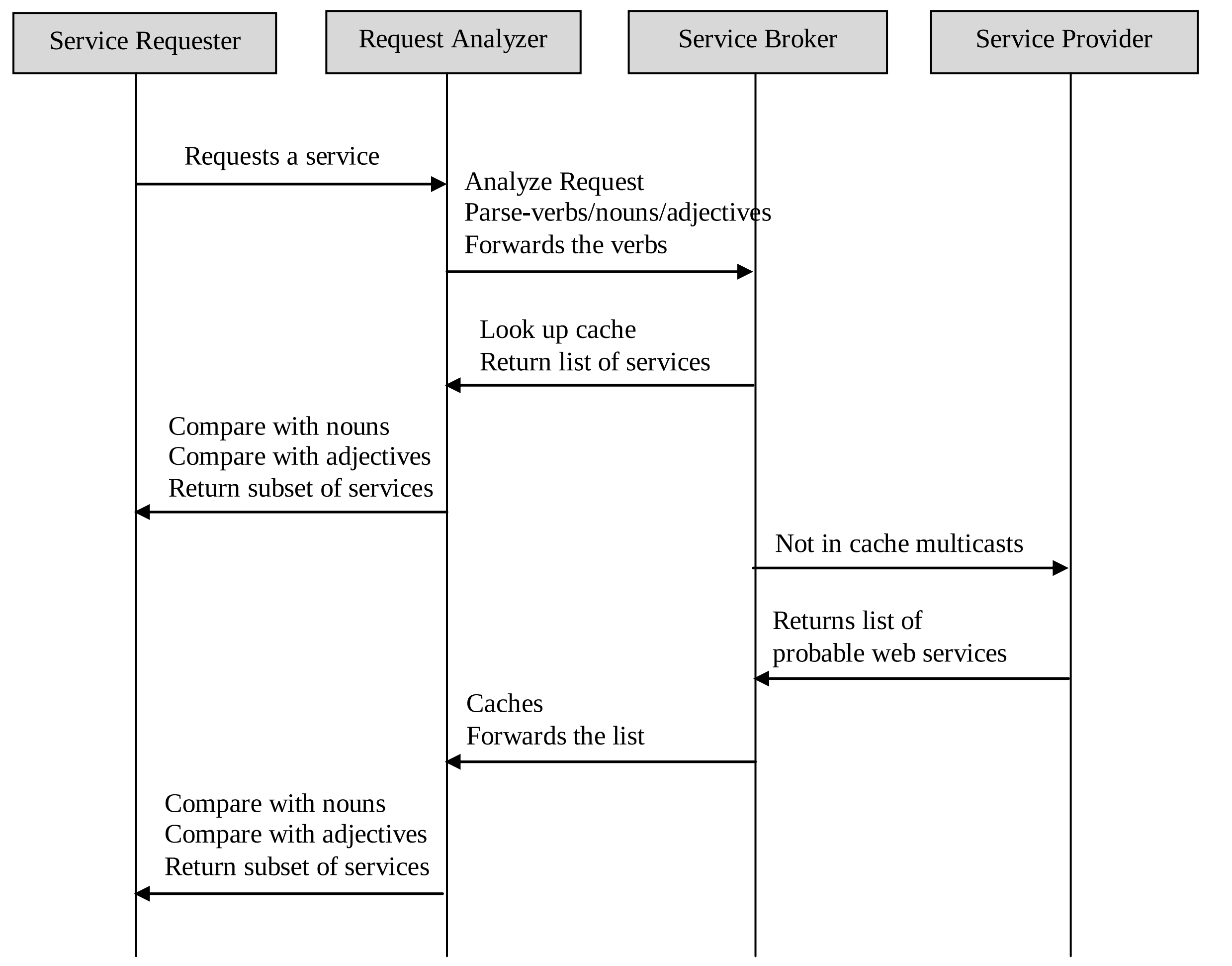

Figure 3: Interaction between various layers of architecture (discovery)

4. Comparison of our framework with other agent based models

Table 1. No. of Message exchanges in different frameworks

Framework	No. of Message Exchanges
LWSDM-with cache	8
LWSDM without cache	6
FIPA based Multi-agent framework	13
SOA 3-tier architecture	6

A: LWSDM with cache
B : LWDM without cache
C: FIPA based Multi-agent framework
D: SOA 3-tier architecture

The plot shows that our architecture reduces the no. of message exchanges to approx 60% from C. Also, even though we have added a new layer for the pre-processing of web requests to the basic SOA 3-tier architecture yet we are able to achieve equal or nearly equal message exchanges to the basic SOA 3-tier architecture, thereby not creating a huge impact on the message exchange cost and network traffic.

5. Conclusion and Future Work

The service consumer is an application, service, or some other software module that requires a service. It initiates the locating of the service in the registry, binding to the service over a transport link, and executing the service function. This architecture will help to efficiently discover a set of most appropriate service for the service consumer. Exploiting the linguistic semantics instead of plain key word based search will make the selection of web services more appropriate. The cache based approach will enable an economic bound on the time and cost of a fresh web search. In addition it will also serve as a primary look up for the RA for building the trust database. The Reputation and service type info shall serve as an aid for the requester to select a suitable and trustworthy web service in future.. The architecture helps to reduce the burden of a novice requester of placing the requests in XML or SOAP formats and also offloads the message handling and pre-processing functionalities of the broker. The open source tool SharpNLP is used to extract parts of speech of the request query. It provides a collection of Natural Language Processing tools that provides facilities like a sentence splitter, a tokenizer, a parts of speech tagger, a parser. It also provides an interface to the WordNet lexical database. The data stores can be implemented using any RDBMS. The scope of future work remains in presenting a complete implementation of the proposed work and enhancing the strength in the trust aspects of the used web services.

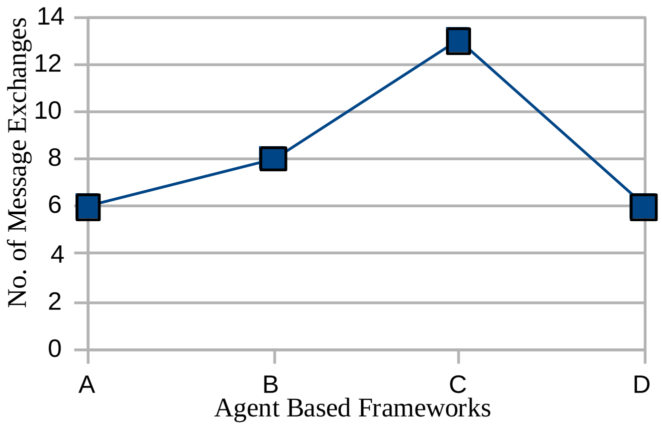

Figure 4: Comparison of no. of message exchanges in various agent based frameworks

6. References

[1] San-Yih Hwang, Ee-Peng Lim, Chien-Hsiang Lee, Cheng-Hung Chen, On Composing a Reliable Composite Web Service: A Study of Dynamic Web Service Selection 2007 IEEE International conference on Web Services

[2] Mike P. Papazoglou, Willem-Jan van den Heuvel Service Oriented Architectures: approaches, technologies and research Issues, The VLDB Journal (2007), Springer-Verlag Publication

[3] Wuling Ren, Zhujun Xu, A New Web Service Discovery Method Based on Semantic, IEEE 2008 Workshop on Power Electronics and Intelligent Transportation System

[4] Jiangang Ma, Yanchun Zhang, Jing He, Efficiently finding Web Services Using a Clustering Semantic Approach, CSSSIA 2008, Copyright ACM

[5] Georgios Meditskos and Nick Bassiliades, Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S, IEEE Journal Publication, Apr 2009

[6] Eyhab Al-Masri and Qusay H. Mahmoud, Discovering the Best Web Service: A Neural Network-based Solution, IEEE Conference on Systems, Man and Cybernetics, October 2009

[7] Ranjit Singh, Shakti Mishra, Dr. D. S Kushwaha, An Efficient Asynchronous Mobile Web Service Framework, December 2009 SIGSOFT Software Engineering Notes , Volume 34 Issue 6, Publisher ACM

[8] Azadeh Ghari Neiat, Mehran Mohsenzadeh, Sajjad Haj Shavalady, Amir Masoud Rahmani, A new approach for Semantic Web Services Discovery and Propagation based on Agents, April 2009 IEEE International Conference on Networking and Services

[9] Che Shin Yeo and Rajkumar Buyya, Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility, 2005, Cluster Computing, IEEE International Conference

[10] Sun Microsystems Utility Computing, http://www.sun.com/service/utility, May 2005