ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

Web Services Manager Model

D. R. Ingle, Dr. B. B. Meshram

Abstract— ***E-customers are highly attracted towards web services in E-business Environment. The rapidly emerging technology of Web services paves a new cost-effective way of engineering software to quickly develop and deploy Web applications by dynamically integrating other independently developed Web-service components to conduct new business transactions. Customers are highly interested to use web services. Millions of users can participate and collaborate for their own interests and benefits. Discovery for the same is very important issue. In this paper we have done the extenive literature survey on the service discovery requirements from the service consumer's perspective and studied a conceptual Model of homogeneous Web service communities. The homogeneous service community contains two types of discovery: the search of similar operations and that of composite operations and dynamic Web Service Selection Strategies, Second, we describe a similarity measurement model for Web services by leveraging the metadata from WSDL, and graph-based algorithm to support both of the two discovery types. We studied Clustering mining for Discovery of homogenous and distinct web services. We have done the literature survey to maintain the security while dealing with web service by using some WS security standards.***

Index Terms— **WSDL –Web services Descriptive Language WS – Web Services. UDDI –Universal Description Discovery and Integration RSS – Rich Site Summary.**

I. INTRODUCTION

Web Services Manager (WSM) is component of SOA Suite. This component is used to secure Web Services and to monitor activities performed on protected Web Services. The Web keeps rapidly growing in recent years. Current Web has been a "user-centric" environment where millions of users can participate and collaborate for their own interests and benefits .The services computing paradigm together with the proliferation of Web services make the Internet as a huge resource library, and millions of users can participate and create more value-added services by means of service discovery and composition [1]. WEB service is defined as a software system designed to support interoperable machine-to-machine interaction over a network .Put in another way, Web services provide a framework for system integration, independent of programming language and operating system. Web services are widely deployed in current distributed systems and have become the technology of choice for implementing service-oriented architectures (SOA). In such architectures [2]. Existing SOA technologies, including Universal Description, Discovery, and Integration (UDDI) and service composition languages (such as BPEL4WS), have fostered agile integration by simplifying integration at the communication, data, and business logic layers. Furthermore, by leveraging efforts in semantic Web services, service composition frameworks made a forward step on enabling automated support for service description matching. Service discovery is a significant activity in Services Computing paradigm. Efficient discovery plays a crucial role in conducting further service composition. With the ever increasing number of services over Internet, more and more service consumers (including nonexpert users, Small and Medium Enterprise, and transient business partners of specific opportunities/interests) can participate in the composition activity1 [1]. Meanwhile, a key problem also matters locating the desired services efficiently. Although existing discovery techniques have produced promising results that are certainly useful, they may not well aligned with the needs of Internet-scale environment. First, searching Web services via some public UDDI registries is mainly based on the keywords involved in query and matches them with the Web service descriptions. As the keywords are not able to capture the underlying semantics, they may miss some results and return a lot of irrelevant ones as well. Second, the users would like to specify their requests more precisely rather than just keywords. Actually, searching Web services is searching for the operations offering some functionality, and current discovery usually explores details of the service operations. The service consumers have to browse each returned results in detail and check if they meet their requirements or not. Nevertheless, investigating a single operation usually needs several steps. Hence, the more service providers emerge, the heavier burden it brings to the consumers. Third, the Web services are developed and maintained by their providers. For some reasons, such as the market competitions and cost control policies, the providers may update or remove their Web services at any time. Once the Web services are modified or even no longer available, the service consumers have to repeat the discovery process to find new appropriate services. To best of our knowledge, current discovery approaches cannot deal with the ever-changing Web services. Due to the reasons above, we argue that current service discovery significantly prevents its ubiquitous adoption among Internet users. Various search engines automatically trawl a set of Web pages and classify them into groups. Moreover, the search engine can even retrieve the access path according to the hyperlinks, which looks like the "composition" of the Web pages. Such search manner makes Web search engine widely adopted by the Internet users. Similarly, in the area of Web service discovery, once the consumers drill down all the way and

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

find the Web service inappropriate for some reason, they may prefer being able to find a set of similar operations that takes similar inputs/ outputs to the ones just considered, instead of laboriously browsing them one after another [2]. What's more, they may intend to find operations that can be composed with the current ones being browsed. Therefore, it seems to be reasonable to support search for similar Web service that can do the job of clustering, classification, match-making, and composition. Such manner will be more free and efficient for the consumers to find their desired services. Another promising hint comes from the innovations of Web 2.0 wave. In our investigation, the most prevailing Web 2.0 communication mechanism is not the complex "centralized registry," but lightweight manner such as the RSS With the feeds, the users are able to organize several Web pages such as news or Weblogs to a specific topic/interest and subscribe them. Once the resources are changed or updated, the RSS/Atom will notify their subscribers. Similarly, if Web services can be organized into RSS/Atom feeds and subscribed by the service consumers, it will release the work of locating the ever-changing Web services. On the other hand, since most current Web browsers (such as Microsoft Internet Explorer, Firefox, and Safari) all support RSS/Atom, it is then feasible to provide a universal and convenient channel for the service consumers, which allows them to easily locate desired Web services and further participate in the service composition. XML based SOAP messages form the basis for exchanging information between entities in Web services systems. The information contained within these SOAP messages may be subject to both confidentiality and integrity requirements. XML Signature and XML Encryption are used to provide integrity and confidentiality respectively. Although these two standards are based on digital signatures and encryption, none of them define any new cryptographic algorithms. Instead, XML Signature and XML Encryption define how to apply well established digital signature/encryption algorithms to XML.
This includes:
• A standardized way to represent signatures, encrypted data, and information about the associated key(s) in XML, independent of whether the signed/encrypted resource is an XML resource or not.
• The possibility to sign and/or encrypt selected parts of an XML document.
• The means to transform two logically equivalent XML documents, but with syntactic differences, into the same physical representation. This is referred to as canonicalization. As both XML Signature and XML Encryption rely on the use of cryptographic keys, key management is a prerequisite for their effective use on a larger scale. Therefore, the XML Key Management Specification (XKMS) was created to be suitable for use in combination with XML Signature and XML. Our aims to provide simpler and more efficient Web service discovery. In this paper we are going provide the best discovery methods of web services that e-customer look for. At the same to we are trying to maintain the security level for the same. Section 2 deals with literature survey on web services, Section 3deals with securing Web Services that contains: Discovery of Homogeneous Web Service, Clustering Task and Security Implementation. Section 4 deals with performance management and section 5 conclude the results.

II. WEB SERVICES

The Web service discovery is a hot research topic in the past a few years. Zhang et al. indicate that the service discovery and composition play the crucial role in the area of services computing [1]. To the service consumers, finding similar Web services and aggregating them in a universal access channel is a key requirement. There are some important research topics related to this issue. We classify the current discovery approaches into two categories: the syntactic based discovery, which involves the techniques of UDDI based search, text document search, schema matching, and software component matching; and semantic-based discovery, which is mainly based on ontology. UDDI-based search. In the initial Web services architecture, UDDI works as the broker to register Web services into corresponding categories. However, to the best of our knowledge, the public UDDI never works as expected. In January 2006, the shutdown of UDDI Business Registry (UBR) operated collaboratively by Microsoft, SAP, and IBM has confirmed the intrinsic problem of the Internet-scale registry-based service discovery. The core reason of public UBR's failure is that the registry-based mechanism is "too complex" for the consumers. UDDI is mainly based on keyword search, which may bring several irrelevant results so that the consumers have to do the "view-select-request" process several times. It is too overwhelming for the consumers to simply get their desired services. Moreover, once the discovered services are no longer available, the discovery process has to be restarted. Thus, we cannot expect that these service consumers can utilize UDDI for service provisioning. The fact of UBR's shutdown has demonstrated that the "Internet-scale" public UDDI cannot be adopted by the huge number of Internet users. In our work, we investigate the service discovery problem from the service consumer's perspective and propose an approach to clustering the homogeneous Web services. It alleviates the consumers from tedious and time-consuming discovery step. With a much easier and universal channel (RSS/Atom) for the service consumers, they are able to subscribe and organize Web services just like Web pages, and track the updates and changes by means of service feeds. Text document search. As Web services are specified in an XML document with an accessible URL, the keyword based text document search is an intuitive approach. In IR community, document matching and classification is a long-standing topic and widely use in most search engines. Due to the fact that the great success of search engines promotes the Web-related search very much, it might be a natural idea to employ the current search techniques for similar Web service discovery. However, most of current information retrieval models are designed for Web pages crawlers and may not work well for Web service discovery due to some key reasons. First, Web pages may contain long textual information. However, Web services have very brief syntactic descriptions (from WSDL files). The lack of textual information makes keyword-based search models unable to filter irrelevant search results, and therefore, become very primitive means for effectively discovering Web services. Second, Web pages primarily contain plain text structures that allow search engines to take advantages of information retrieval models like TF/IDF. However, Web services contain much more complex structure with very little text descriptions provided either on UBRs or service interfaces. It then makes the dependency on information basic retrieval techniques very infeasible. Third, Web pages are described using standard HTML with predefined set of tags. However, Web service definitions are not fully standard as they are developed by independent vendors. Web service interface information such as message names, operation, and parameter names within Web services can vary significantly which makes the finding of any trends, relationships, or patterns within them very difficult and requires excessive domain knowledge in XML schemas and namespaces [3]. Therefore, the current text document search approaches are insufficient in the Web service context. Schema matching. In the database community, the problem of automatically matching schemas investigates the clues of underlying semantics from the schema structure and suggests the matches based on them [18], [19]. In the Web service discovery, schema matching is also employed. In [17], the authors proposed an approach to measuring the similarity between two Web services based on their Tree Edit Distance. However, we argue that there is a big obstacle to apply schema matching to Web service discovery: the operations in a Web service are typically much more loosely related to each other than the tables defined in a schema, and each Web service in isolation has much less information than a schema. Hence, it will be difficult to retrieve the underlying semantics from the schema of WSDL. In current WSDL files, the corresponding information is not available. To the best of our knowledge, the WSDL V 2.0 will adopt the semantic annotations for the data types and specifications. Ontology-based discovery. A very important direction of current Web services research is the semantic Web services group, such as OWL-S[24], WSDL-S [25], and SWSO[26]. It means that it will generate more explicit semantic categorization. In fact, tags are currently very popular descriptions, and can be obtained from those social networking Websites such as del.icio.us and flickr. Certainly, to realize the descriptions of service by tags, it still requires slight and more careful and consistent design, which is beyond the scope of this paper. In our opinion, we prefer relying on lightweight semantic metadata annotations by making use of tags, folksonomies, and simple taxonomies to describe the semantics of services [14]. The use of tag-based descriptions greatly simplifies the users, compared to the much heavier ontology-based approaches, such as OWL-S. To the best of our knowledge, we have found that some recent works [21],[22]. Another important related work worth mentioning is Woogle [2], a Web service search engine developed by the University of Washington. Woogle employs an unsupervised approach to retrieve the underlying semantics from WSDL and measure the similarity between operations and input/output. Similar to woogle, our approach adopts the semantic clustering algorithm to generate the meaningful concepts. As the concepts clustering results significantly impact the similarity measurement, we need to consider some improvement. Woogle provides a technique to split and merge the clusters by considering the cohesion and correlation. It can remove some terms and improve the clustering results. However, the Web services are developed by independent providers, the parameter naming heavily relies on the developer's personal whim. Once a term in cluster A is associated with more than half of the terms in the cluster B, these two clusters will be merged by such technique. Therefore, in the phase of removing the noise terms, we apply the matching score by employing some taxonomy (use social folksonomy in our experiment), while woogle just processes by measuring the co-occurrence-based association rules.

III. SECURING WEB SERVICES

This section contains three units:
1. Discovery of Homogeneous Web Service
2. Clustering Task
3. Security Implementation

In first part we are identifying the homogeneous & distinct web services by using mining algorithm by measuring the co-occurrence of terms from the various sources [1]. In second unit we use clustering algorithm to cluster the homogeneous web services [22] and in last unit we identify the requirements to maintain the security [12].

A. Discovery of Homogeneous Web Service

In this unit service consumer deals with service community, service container is nothing but the local database which is use to store all information related web services classification and indexing is used to determine the similarity with the help of semantics. It employs a crawler to retrieve the WSDL files from the service registries and store them into a local database as the metadata. The component Metadata Parser analyzes each WSDL file, filter the irrelevant information, and retrieve terms from the input/output data types as shown in fig. 1.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

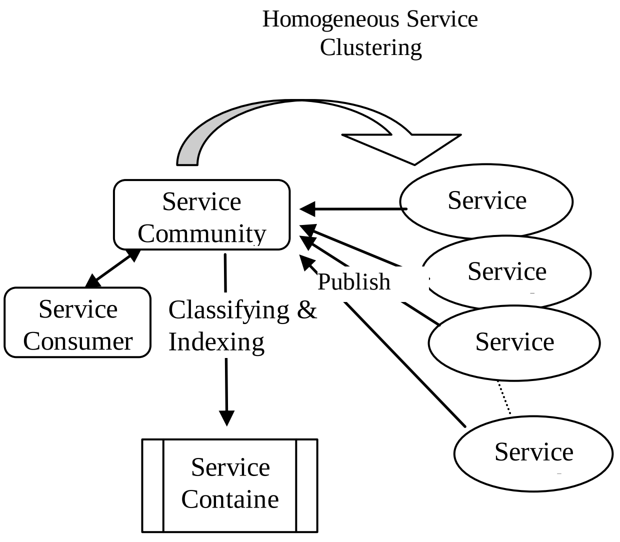

Fig. 1 Service Container based discovery

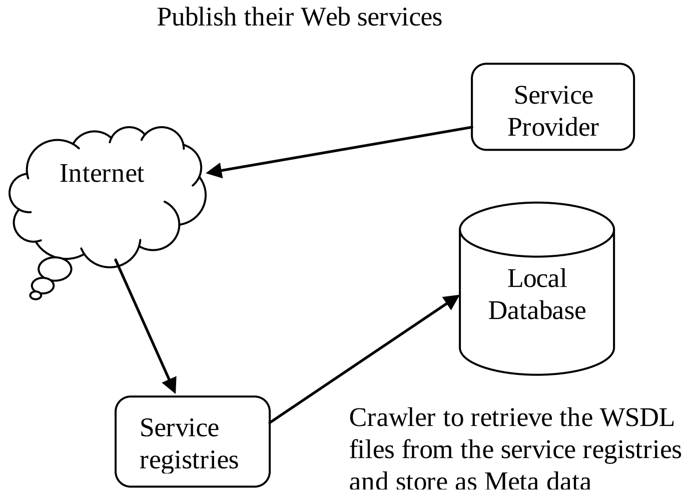
Publish their Web services
Service Provider
Internet
Local Database
Service registries
Crawler to retrieve the WSDL files from the service registries and store as Meta data

Fig. 2. Storage of metadata

The Clustering Agent clusters the terms into concepts, and then, finds the corresponding Web operations as well as the potentially compatible ones. These operations are maintained in a graph, so we need to make index of them with the Web Services they belong to. As discussed before, considering the usability of the users, we adopt the Atom feeds (instead of popular RSS, for Atom is more suitable to process structured XML than RSS), which is widely used for news aggregation and subscription. In Service Container, the Feed Manager component attaches the metadata to the Atom feeds and indexes the feeds to the corresponding Web services by using the standard Atom publish protocol as shown in fig. 2. Here, it is necessary to make the binding between the Atom feeds and WSDL, to ensure the consistency with the original Atom syntax and semantics. To the best of knowledge, in [20] and [24], the authors explicitly integrated atom feeds to a service description, including mapping the metadata of WSDL to the Atom feeds and the feed publish protocol over HTTP. This provides us very useful information in our implementation. We briefly describe the main design ideas. Due to the functionalities that Service container provides, we adopt two feed types. The first one is the Web service entry, which is used to subscribe exactly one Web service. This element represents Web service by binding the metadata of the Web service to an Atom feed. For example, the "Atom Summary" is attached by the useful metadata from WSDL, including the textual description of input/output and operation, which can be used for indexing the Web services. The second feed type is the topic feed, which is used for subscribing a homogeneous Web services (exactly their operations). The topic feed aggregates a set of homogeneous Web services into a group. The topic feed contains the metadata for a list of Web services, each of which corresponds to a service entry. The relationship between service topic feed and service entry is consistent with that defined in Atom specification. Service entry and topic feed correspond to the generic Atom feed and entry. Each entry in the feed is mapped to a single Web services under a particular. First, the users would like to get the set of "single" Web services with similar functionalities. To the best of our knowledge, the functionalities offered by a Web service are usually reflected by its operation. For example, the weather report services may provide operations such as "GetWeatherByZipCode" or "Get Temperature." Therefore, the problem can be viewed as "searching for similar operations." Intuitively, the operations are similar if they have similar inputs, generate similar outputs, and the relationships between the inputs and outputs are also similar [2]. Second, if no single service operation is qualified for the request, the users may also want to retrieve a sequence of operations that can be composed together. It means that the outputs generated by one service can be accepted as the inputs of another service. For example, suppose two Web services S1 and S2: S1 is an "Address Querying" service which can output the city name according to the given zip code (e.g., the output is "Zip Code To City"), and S2 is a "Weather Forecasting" service which can return the weather forecast by the given city name (e.g., the input is "City," "State"). S1 and S2 can be composed together in case that the city name is unknown. Such process can iteratively proceed to construct "operation hyperlink" until the desired result is fulfilled. Both of the two discovery types mentioned above are essentially related to the similarity measurement of Web service operations, inputs, and outputs. Like traditional clustering approaches for Web pages or topics, we are going to employ similarity measurement to retrieve the homogeneous Web services. Then, some technical challenges need to be solved. As is known to all, semantics means crucially to determine the similarity. But in current WSDL specification, neither the textual descriptions of Web services and their operations nor the names of input/output parameters completely convey the underlying semantics of

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

the service operation. Therefore, searching for similar Web services is much more challenging. To efficiently match the inputs/ outputs of Web service operations, it is then important to get their underlying semantics. By investigating the metadata from the WSDL structure. Finally, as Service container is supposed to be an aggregator of a group of homogeneous services, such organization style is similar to the news group that can be subscribed with RSS or Atom. Moreover, with the RSS/Atom, the consumers will be notified once their subscribed Web services are changed. Therefore, we need to bind the WSDL to the RSS/Atom feeds while considering the semantic consistency. Particularly, we find that the current Atom specification has already defined some useful elements for Web service discovery and subscription. For example, each Atom entry has a unique ID, while a Web service has also one to identify itself while regardless the version, location, and invocation information. So, the entry ID is useful when subscribing the "service feed." Another example is "atom link," which is used for the entry and feed. It explicitly defines a mechanism which can flexibly attach some customized metadata into RSS. Thus, when the entry is a WSDL file or OWL-S represented by RDF, we can then import several "Atom link" to describe the MIME types. The service consumer can retrieve these data in a standard manner via HTTP. Our approach tries to combine multiple sources of evidences to determine the similarity between Web services. We describe a mining algorithm that clusters metadata (including input/ output names) from a collection of Web services into some semantically meaningful concepts. By comparing the concepts, they belong to, and considering the similarity of the descriptions of the operations and the entire Web services, we can have a good similarity measurement. Besides the similarity measurement, we may need a search model. Within the Service container, we hold two types of search: single similar operations and compatible operation sequences. Thus, the search model is expected to be able to process both of the two types, and promise high efficiency. we merge these two types by employing an algorithm based on a Directed Graph/Huffman code. We make each operation as a vertex, maintain the composition opportunities as directed edges, and assign the weight of the edge with similarity matching score between inputs and outputs. Then, the search for the single similar operations is transformed to the traversal of all vertexes, and the search of compatible operations is transformed to find the corresponding paths. [1]

B. *Clustering Task*

A Web service is described in an XML-based document, called WSDL. The WSDL specifies the service implementation definition and service interface definition the service implementation definition describes how a service interface is implemented by a given service provider, and the service interface definition contains the business category information and interface specifications that are registered as UDDI tModels. Input message and one output message. Note there is a set of operations in a WSDL. Input/Output: 8input 2 Mð8output 2 MÞ is a message m. Each input and output contains a set of parameters for an operation defined by the message element and the type element used in the message (for representing the complex data types). . a message m 2 M has optionally iðı _ 1Þ parts, represented as m ¼ fd1; d2; . . . ; dkg, where dj 2 D; 1 _ j _ k. From the definition, regardless of the invocation information in WSDL that is useless for similarity matching, such as the binding and the port, we can identify three types of metadata from WSDL. First, we note the plain textual descriptions, which describe the general kind of service that is offered, for example, service related to "weather forecasting" or "travel agency." Second, we note the domain of the operation that captures the purposed functionality, such as "Get Weather ByZipCode," "Search Book," or "Query Airplane Timetable." Finally, we find the data type deriving from the input/ output. The data types do not relate to the low-level encoding issues such as integer or string, but to the semantic meanings such as "weather," "zip code," etc.

a) Estimating the Parameters by Huffman Code

In terms of similarity measurement, the service descriptions can be easily determined by the traditional Term Frequency and Inverse Document Frequency (TF/IDF) methods. However, the similarity of operations and inputs/outputs cannot be determined. On one hand, the parameter naming is mostly dependent on the service provider/developer's personal habit. Hence, parameters tend to be highly varied given the use of synonyms, hypernyms, and different naming rules. On the other hand, inputs/outputs typically have very few parameters, and the associated WSDL files rarely provide rich description. We try to explore the underlying semantics of the inputs/outputs in addition to their textual descriptions. First, an intuitive heuristic is that the parameter names, which are specified in the inputs/ outputs and operations, are often combined as a sequence of several terms. Take the parameter "GetWeatherByZipcode," for example, the terms are specified by their first letter capitalized {Get, Weather, By, Zip code}.We cluster these terms into several concepts. In our opinion, considering the terms with the concepts they belong to May significantly improve the similarity measurement. For example, given the two outputs {weather} and {temperature, humidity}, they cannot be considered to be similar just by checking with their names. But these terms are all related to the concept of "weather," they should be similar data types.

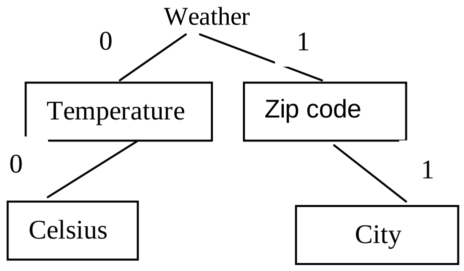

Fig 3. Temperature details

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

As shown in above figure 3: Huffman code can be used to calculate the similarities for input. Supposed the requested input parameter is "GetWeatherByZipcode," we can find similar web services which matches to our input by assigning o and distinct by assigning 1 and add them into the local database (assign o & 1 to the clusters).We can consider the term Weather according to that we can find the related clusters in our database It is save in database when we want the data related temperature then it will then 0 only that is using BFS.

$$\text{Weather} \longrightarrow \text{Temperature} \longrightarrow \text{Celsius}$$

For Clustering the similar web services we used Clustering algorithm K- Means Algorithm When clustering metadata residing in the input/output data types into several meaningful semantic concepts, we intuitively consider the words co-occurrence. A common sense heuristic is that the words tend to express the same semantic concept if they often occur together [17]. In other words, similar data types tend to be named by similar names, and/or belong to messages and operations that are similarly named. Therefore, we can then exploit the conditional probabilities of occurrence.

K- Means Algorithm:

The k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, k. This algorithm has been discovered by several researchers across different disciplines. This algorithm would be used in the project for choosing the best investor for a queried investor and vice-versa. In statistics and data mining, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. This results into a partitioning of the data space into Voronoi cells. However k-means clustering tends to find clusters of comparable spatial extend, while the expectation-maximization mechanism allows clusters to have different shapes. Standard algorithm: The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the k-means algorithm; it is also referred to as Lloyd's algorithm, particularly in the computer science community. Given an initial set of k means m1 (1),…,mk(1) (see below), the algorithm proceeds by alternating between two steps. Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).

Update step: Calculate the new means to be

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \big\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \big\| \leq \big\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \big\| \text{ for all } i = 1, \ldots, k \right\}$$

Update step: Calculate the new means to be the centroid of the observations in the cluster.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{(t)} \mathbf{x}_j$$

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \big\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \big\| \leq \big\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \big\| \text{ for all } i = 1, \ldots, k \right\}$$

is deemed to have converged when the assignments no longer change. Commonly used initialization methods are Forgy and Random Partition. The Forgy method randomly chooses k observations from the data set and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the Update step, thus computing the initial means to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al., the Random Partition method is generally preferable. b) Predicting the Similarity: In previous section, it clustered the concept as the baseline to measure the similarity for inputs/outputs. Now, we will compute similarity for the Web service operations. As defined in Section 3.1, an operation op is a three-tuple vector op ¼< nop; input; output > , then given two operations opi; opj, we can determine the similarity by combining the similarity of each individual elements, respectively. First, we estimate the similarity of the text description of operation and the Web services the operation belongs to (represented by Nw), it can be achieved by employing the traditional TF/IDF measurement. Next, we estimate the similarity of the input and output by considering the underlying semantics the input/output parameters cover. Formally, it analyze the input as a three-tuple vector input ¼< nin; Ci > (similarly, the output can be represented in the form of output ¼< nout; Co >), where nin is the text description of input names and Ci is the concept that associates with nin. Then, the similarity of input can be done in the following two steps:First, we evaluate the similarity of the descriptions of input names by TF/IDF. . Second, we split nin into a set of terms. Note that we should filter the terms related to outputs (such as "Zip Code" in the input "City Name By ZipCode"). c) Prototype Architecture we assume that the service providers can publish their Web services on the Internet as usual. Then, we employ a crawler to retrieve the WSDL files from the service registries and store them into a local database as the metadata. The component Metadata Parser analyzes each WSDL file, filter the irrelevant information, and retrieve terms from the input/output data types. Using the similarity measurement approach described in Section 3, the Clustering Agent clusters the terms into concepts, and then, finds the corresponding Web operations as well as the potentially compatible ones. These operations are maintained in a graph, so we need to make index of them with the Web services they belong to. Considering the usability of the users, we adopt the Atom feeds (instead of popular RSS, for Atom is more suitable to process structured XML than RSS), which is widely used for news aggregation and subscription. In Service Container, the Feed Manager component attaches the metadata to the Atom feeds and indexes the feeds to the corresponding Web services by using the standard Atom publish protocol For example, the "Atom Summary" is attached by the useful metadata from Here, it is necessary tomakethe binding between the Atom feeds and WSDL, to ensure the consistency with the original Atom syntax and semantics. To the best of knowledge [20] and [24], the author's explicitly integrated atom feeds to a service description, including mapping the

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

metadata of WSDL to the Atom feeds and the feed publish protocol over HTTP. This provides us very useful information in our implementation. We briefly describe the main design ideas. Due to the functionalities that Service Container provides, we adopt two feed types. The first one is the Web service entry, which is used to subscribe exactly one Web service. This element represents Web service by binding the metadata of the Web service to an Atom feed. We list the some core mappings .WSDL, including the textual description of input/output and operation, which can be used for indexing the Web services.

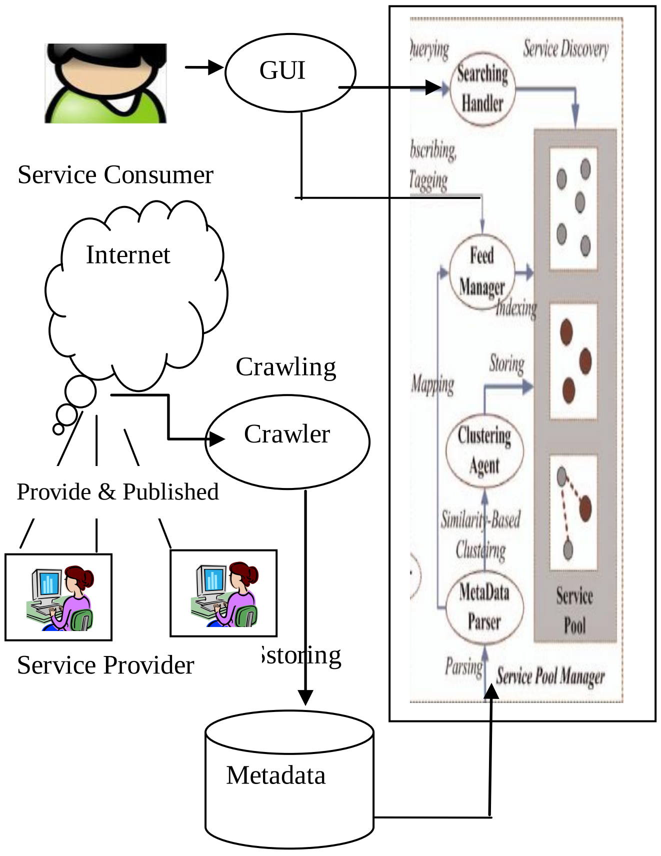

Fig. 4 Prototype Architecture

The second feed type is the topic feed, which is used for subscribing a homogeneous Web services (exactly their operations). The topic feed aggregates a set of homogeneous Web services into a group. The topic feed contains the metadata for a list of Web services, each of which corresponds to a service entry. The relationship between service topic feed and service entry is consistent with that defined in Atom specification. Service entry and topic feed correspond to the generic Atom feed and entry. Each entry in the feed is mapped to a single Web services under a particular topic as shown in fig. 4.

d) Dynamic Web Service Selection Strategies

AR-based Selection and Composability and AR (CAR)-based Selection strategies. AR-based Selection Strategy. The rationale behind the AR-based selection strategy is to select an atomic WS for each incoming operation of the composite WS so as to achieve maximum reliability. At runtime, when an incoming operation arrives at a configuration, we first sort the candidate WS operations in no increasing order of the products of their reliabilities and aggregated reliabilities of the destination configurations. These WS operations are tried one at time in the order until one gets successfully executed. This strategy, called AR-based selection shown as below

```
       AR_based_selection($F$: a composition) {
       // $F = (\Sigma_c, S_c, s_{c,0}, \delta_c, F_c)$
1      $A[]$ = Final_Agg_Reliability($F$);
2      $c = s_{c,0}$;
3      wait($o$);
4      repeat
5        $Candidates = \{W.o: W.o \in \Sigma_c, \delta_c(c, W.o)$ is defined$\}$
6        forall $W.o \in Candidates$ listed in non-increasing
7          order of $R(W.o) \cdot A[\delta_c(c, W.o)]$ do {
8              if executing $W.o$ is successful then
9                  $c = \delta_c(c, W.o)$
10                 break 13;
11         }
12         report(" failure"); exit;
13         wait($o$);
14     until $o$ = 'end of WS';
15     }
```

```
       CAR_based_selection($F$: a composition) {
       // $F = (\Sigma_c, S_c, s_{c,0}, \delta_c, F_c)$
1        $A[]$ = Final_Agg_Reliability($F$);
2        $c = s_{c,0}$;
3        wait($o$);
4        repeat
5          $All = \{W.o: W.o \in \Sigma_c, \delta_c(c, W.o)$ is defined$\}$
6          $Composable = \{W.o: W.o \in All, \delta_c(c, W.o)$ is com-
               posable$\}$
7         if $Composable \neq \varnothing$ then $Candidates = Composable$
8          else $Candidates = All$;
9          forall $W.o \in Candidates$ listed in non-increasing
                 order of $R(W.o) \cdot A[\delta_c(c, W.o)]$ do {
10             if executing $W.o$ is successful then
11                $c = \delta_c(c, W.o)$
12                break 15;
13            }
14          report("failure"); exit;
15          wait($o$);
16       until $o$ = 'end of WS';
17     }
```

CAR-based Selection Strategy. This selection strategy considers aggregated reliabilities as well as the composabilities of configurations in selecting atomic WSs. It considers composable configurations whenever possible in choosing a delegated operation. Between two WSs whose destination configurations are composable (or otherwise), the strategy prefers the one with higher product of its reliability and the AR of the destination configuration.

C. *Security Implementation*

XML based SOAP messages form the basis for exchanging information between entities in Web services systems. The information contained within these SOAP messages may be subject to both confidentiality and integrity requirements. Although mechanisms at lower layers may provide end-to-end security, these lower layer mechanisms are often insufficient. This is due to the fact that a SOAP message may be subject to processing and even modification (e.g., removal/insertion of a SOAP header) at intermediary nodes. The result being that the end-to-end security provided by lower layer mechanisms Relying on lower layers for end-to-end security may also cause problems if a message is to pass through various networks utilizing different transport protocols. Furthermore, security at the XML level has the advantage of enabling confidentiality and source integrity to be maintained also during storage at the receiving node(s).XML Signature and XML Encryption are

used to provide integrity and confidentiality respectively. Although these two standards are based on digital signatures and encryption, none of them define any new cryptographic algorithms. Instead, XML Signature and XML Encryption define how to apply well established digital signature/encryption algorithms to XML.
This includes:
• A standardized way to represent signatures, encrypted data, and information about the associated key(s) in XML, independent of whether the signed/encrypted resource is an XML resource or not.
• The possibility to sign and/or encrypt selected parts of an XML document.
• The means to transform two logically equivalent XML documents, but with syntactic differences, into the same physical representation. This is referred to as canonicalization. In order to be able to verify the signature of an XML resource that has had its representation changed, but still has the same logical meaning, it is essential that canonicalization is performed as part of the XML signature creation and verification processes. As both XML Signature and XML Encryption rely on the use of cryptographic keys, key management is a prerequisite for their effective use on a larger scale. Therefore, the XML Key Management Specification (XKMS) was created to be suitable for use in combination with XML Signature and XML. The Web Services Security (WSS) specifications aim to provide a framework for building secure Web services using SOAP, and consist of a core specification and several additional profiles. The core specification, the Web Services Security: SOAP Message Security specification [14] (WSSecurity for short), defines a security header for use within SOAP messages and defines how this security header can be used to provide confidentiality and integrity to SOAP messages. XML Encryption is utilized to provide confidentiality, while message integrity is provided through the use of XML Signature. Using these mechanisms, SOAP message body elements, selected headers, or any combination thereof may be signed and/or encrypted; potentially using different signatures and encryptions for different SOAP roles (i.e., different intermediaries and ultimate receiver(s)). Recall (from Section II) that because SOAP message headers may be subject to processing and modification by SOAP intermediaries, lower layer security mechanisms such as SSL/TLS are often insufficient to ensure end-to-end integrity and confidentiality for SOAP messages. For such messages, the functionality provided by WS-Security is essential if confidentiality and/or integrity are required. These are the X.509 certificate token profile [22], the Rights Expression Language (REL) token profile the Kerberos token profile [24], the Username Token Profile [25], and the SAML token profile [9]. There is also a WSS: SOAP Messages with Attachments (SwA) Profile, which is applicable to SOAP 1.1 but not to SOAP 1.2. We can use SHA-1 algorithm for implementing the security [20]. **SHA-1-**algorithm or rather function would be helpful in our project to maintain the security of information (contact details and passwords), so that even our integral parts such as administrator wouldn't be able to view such information. Description: SHA-1 produces a 160-bit message digest based on principles similar to those used by Ronald L. Rivest of MIT in the design of the MD4 and MD5 message digest algorithms, but has a more conservative design. The original specification of the algorithm was published in 1993 as the Secure Hash Standard, FIPS PUB 180, by US government standards agency NIST (National Institute of Standards and Technology). This version is now often referred to as SHA-0. One iteration within the SHA-1 compression function: A, B, C, D and E are 32-bit words of the state; F is a nonlinear function that varies; n denotes a left bit rotation by n places; n varies for each operation;Wt is the expanded message word of round t; Kt is the round constant of round t; SHA-1 differs from SHA-0 only by a single bitwise rotation the message schedule of its compression function; this was done, according to NSA, to correct a flaw in the original algorithm which reduced its cryptographic security. However, NSA did not provide any further explanation or identify the flaw that was corrected. Weaknesses have subsequently been reported in both SHA-0 and SHA-1. SHA-1 appears to provide greater resistance to attacks, supporting the NSA's assertion that the change increased the security.
1) *The Username Token Profile:* The Username Token profile [25] specifies how the Username Token can be used as a means to identify a requester by username. A password, or some sort of shared secret constituting a password equivalent, may also be included. Passwords may be included in their original form or as a SHA-1 digest. In order to prevent replay attacks, The SHA-1 password digest is to be calculated over the nonce, timestamp, and password, thus, both the sender and the receiver need to know the plaintext password or password equivalent.
2) The X.509 Certificate Token Profile: The X.509 certificate token profile [22] defines how to include X.509 certificates in SOAP messages. Such certificate tokens may be used to validate the public key used for authenticating the message or to specify the public key, which was used to encrypt the message (or more commonly to convey the secret key used to encrypt the message). When the X.509 certificate is used to authenticate the sender, ownership of the certificate token is proved by signing the message using the corresponding private key.
3) The Rights Expression Language (REL) Token Profile: The Rights Expression Language (REL) token profile [23] defines how to include ISO/IEX 21000-5 Rights Expressions in SOAP messages. In the context of XML and Web services, the Rights Expression Language is also known as the XML Rights Management Language (XrML). Although a technical committee was formed within OASIS in order to standardize XrML, this committee was disbanded before reaching an agreement on a standard. *The SAML Token Profile:* The SAML token profile [26] defines how to include SAML assertions within security headers and how to reference these assertions from within the SOAP message. A binding between a SAML token and the SOAP message (and its sender) can be created by signing the message with a key specified within the SAML assertion.[2] Alternatively, an attesting entity that the

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

receiver trusts may vouch for the message being sent on behalf of the subject for whom the assertion statements apply. In this latter case, the attesting entity must ensure the integrity of the vouched for SOAP message (e.g., by applying a digital signature). SAML is discussed in more detail in Section V-B.

5) The Kerberos Token Profile: The Kerberos token profile [24] defines how to attach Kerberos tickets to SOAP messages.

The specification is limited to the Kerberos AP-REQ message [28], allowing a client to authenticate to a service. Like with the X.509 certificate token, ownership of the token is proved by signing the message using the corresponding key. How the AP-REQ is to be obtained is outside the scope of the profile, but such functionality is provided by the Kerberos specification and might also be provided using WS-Trust.

6) The Basic Security Profile: The Web Services Interoperability Organization (WS-I) has also defined another related profile called the Basic Security Profile [26]. This profile provides clarifications, and requirements, on how WS-Security and its associated profiles should be implemented in order to promote interoperability. Because WS-Security makes use of XML Signature and XML Encryption, the Basic Security Profile also applies to XML Signature and XML Encryption when these are used with WS-Security.

IV. PERFORMANCE MANAGEMENT

The system supports multiple classes of web services traffic and allocates server resources dynamically so to maximize the expected value of a given cluster utility function in the face of fluctuating loads. The cluster utility is a function of the performance delivered to the various classes, and this leads to differentiated service. In this paper, we will use the average response time as the performance metric. The management system is transparent: it requires no changes in the client code, the server code, or the network interface between them. The system performs three performance management tasks: resource allocation, load balancing, and server overload protection. We use two nested levels of management. The inner level centers on queuing and scheduling of request messages. The outer level is a feedback control loop that periodically adjusts the scheduling weights and server allocations of the inner level. The feedback controller is based on an approximate first-principles model of the system, with parameters derived from continuous monitoring. We focus on SOAP-based web services [22]. This system allows service providers to offer and manage SLAs for web services. The service provider may offer each web service in different grades, with each grade defining a specific set of performance objective parameters. For example, the Stock Utility service could be offered in either premium or basic grade, with each grade differentiated by performance objective and base price. A prototypical grade will say that the service customers will pay $10 for each month in which they request less than 100 000 transactions and the 95th percentile of the response times is smaller than 5 s, and $5 for each month of slower service. Our management system allocates resources to traffic classes and assumes that each traffic class has a homogeneous service execution time. We introduce the concept of class to separate operations with widely differing execution time characteristics. For example, the Stock Utility service may support the operationsgetQuote () and buy Shares ().

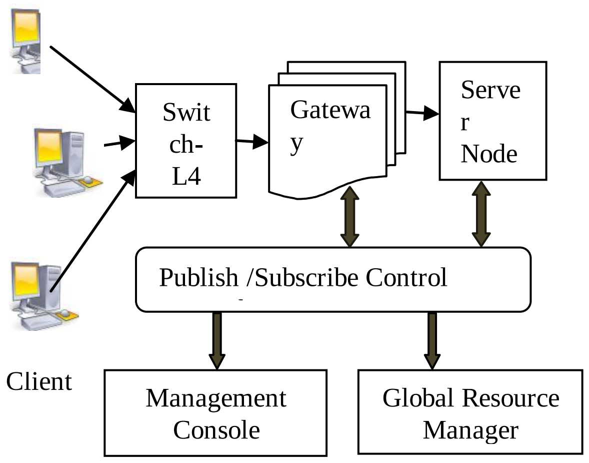

Fig. 5 Performance Management for Clustered Based Web Services.

The fastest execution time for get Quote () could be 10 ms, while the buy Shares () cannot execute faster that 1 s. In such a case, the service provider would map these operations into different classes with different set of response time goals.[22] We also use the concept of class to isolate specific contracts to handle the requests from those customers in a specific way. Figure 5 shows the system architecture. The main components are a set of gateways, a global resource manager, a management console, and a set of server nodes on which we deploy the target web services. We use gateways to execute the logic that controls the request flow, and we use the server nodes to execute the web services logic. Gateway and server nodes are software components. We usually have only one gateway per physical machine and, in general, we have server nodes and gateways on separate machines. The simplest configuration is one gateway and one server node running on the same physical machine. In this paper, all server nodes are homogeneous and that every web service is deployed on each server. We can deal with heterogeneous servers by partitioning them into disjoint pools, where all the servers in a given pool have the same subset of web services deployed, and where the traffic classes are also partitioned among the pools. The servers, gateways, global resource manager, and console share monitoring and control information via a publish/subscribe network [23].

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

A. Gateway: We use gateways to control the amount of server resources allocated to each traffic class. By dynamically changing the amount of resources, we can control the response time experienced by each traffic class. We denote Ng,s with the maximum number of concurrent requests that server executes on behalf of gateway . We also Use Wg,c to describe the minimum number of class requests that all servers will execute on behalf of gateway . We refer to Wg,c as *server shares*. In Section IV, we will describe how we compute Wg,c and Ng,s , while in this section, we describe how gateway enforces the Wg,c and Ng,s constraints. For each gateway , we use Wg and Ng to denote the following:

$$w_g = \sum_{c \in C} w_{g,c}, \qquad N_g = \sum_{s \in S} N_{g,s}$$

Where C and S denote the set of all classes and servers, respectively. We have used Axis [22] to implement all our gateway components, and we have implemented some of the mechanisms using Axis handlers, which are generic interceptors in the stream of message processing. Axis handlers can modify the message, and can communicate out-of-band with each other via an Axis message context associated with each SOAP invocation (request and response) [22]. When a new request arrives a *classification handler* determines the traffic class of the request. The mapping functions use the request metadata (user id, subscriber id, service name, etc.). In our implementation, the classification handler uses the user and SOAP action fields in the HTTP headers as inputs, and reads the mappings from configuration files. We avoid parsing the incoming SOAP request to minimize the overhead. After we classify the requests, we invoke the *queue handler*, which in turn contacts a *queue manager*. The queue manager implements a set of logical FIFO queues one for each class. When the queue handler invokes the queue manager the queue manager suspends the request and adds the request to the logical queue corresponding to the request's class. The queue manager includes a *scheduler* that runs when a specific set of events occurs and selects the next request to execute. The queue manager on g gateway tracks the number of outstanding requests dispatched to each server and makes sure that there are at most Ng requests concurrently executing on all the servers. When the number of concurrently outstanding requests from gateway is smaller than Ng the scheduler selects a new request for execution. The scheduler uses a round-robin scheme. The total length of the round-robin cycle Wg is and the length of class interval is Wg,c .We use a dynamic boundary and work conserving discipline that always selects a nonempty queue if there is at least one. The *dispatch handler* selects a server and sends the request to the server, using a protocol defined by configuration parameter. Our implementation supports SOAP over HTTP and SOAP over JMS. The dispatch handler distributes the requests among the available servers using a simple load balancing discipline, while enforcing the constraint that at most Ng,s requests execute on server concurrently on behalf of gateway . When a request completes its execution, the *response handler* reports to the queue manager the completion of the request's processing. The queue manager uses this information to both keep an accurate count of the number of requests currently executing and to measure performance data such as service time. The gateway functions may be run on dedicated machines, or on each server machine. The second approach has the advantage that it does not require a sizing function to determine how many gateways are needed, and the disadvantage that the server machines are subjected to load beyond that explicitly managed by the gateways.

B. Global Resource Manager and Management Console

The *global resource manager* computes Ng,s the maximum Number of concurrent requests that each server executes on Behalf of each gateway, and it computes Wg,c the minimum Number of class requests that all servers will execute on the behalf of each gateway.

$\sum_{s \in S} N_{g,s}$ Represents the total amount of resources allocated to gateway , while Wg,c is the portion of that dedicated to class . Given these two sets of parameters, a gateway is able to perform WRR scheduling, and load balancing. The global resource manager runs periodically and computes the resource allocation parameters every time interval Γi which we define as the th control horizon. The global resource manager computes Ng,s and Wg,c that each gateway will use during the control horizon using the resource allocation parameters computed in the control horizon Γi-1 as well request and server utilization statistics measured in during Γi-1. The size of the control horizon affects the ability of the global resource manager to respond to rapid changes in the traffic load or response time. On the one hand, when Γi is small, the resource allocation parameters are updated frequently which make the system more adaptive. On the other hand, a larger value of Γ increases the stability of the system. The global resource manager inputs and outputs. In addition to real-time dynamic measurements, the global resource manager uses resource configuration information, and the *cluster utility function*. The cluster utility function consists of as a set of *class utility functions* and a *combining function*. Each class utility function maps the performance for a particular traffic class into a scalar value that encapsulates the business importance of meeting, failing to meet or exceeding the class service level objective. A combining function combines the class utility function into one cluster utility function. We have implemented two combining functions: sum and minimum. Other combining function on the structure of the solution. The global resource manager may assume the responsibility of computing the capacity Ns of each server .Ns represents the maximum number of web services requests that server can execute concurrently. The global resource manager should select Ns to be large enough to efficiently utilize the server's physical resources, but small enough to prevent overload and performance degradation. The global resource manager may use server utilization data to determine the value of Ns. The global resource manager partitions Ns among all gateways and classes. The global resource manager uses Wg,c to describe the minimum number of class c requests that all servers will execute on behalf of gateway . The global resource manager uses a queuing model

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

of the system to predict the performance that each class would experience for each given allocation Wg,c The global resource manager implements a dynamic programming algorithm to find the Wg,c that maximize the cluster utility function. After the global resource manager computes a new set of Wg,c and Ng,s values, it broadcasts them on the control network. Upon receiving the new resource allocation parameters each gateway switches to the new values of Wg,c and Ng,s. The *management console* offers a graphical user interface to the management system. Through this interface the service provider can view and override all the configuration parameters. We also use the console to display the measurements and internal statistics published on the control network. Finally, we can use the console to manually override the control values computed by the global resource manager.

V. CONCLUSION

We have tried to do extensive literature survey to provide efficient service discovery approach from the service consumer's perspective. After studying the discovery requirements, we identify from literature that the service consumers do prefer attaining the similar Web services and the potentially composible Web services, according to their desired inputs and outputs. These services should be organized in a universal access channel, instead of enforcing the users to search and view them individually. These services are known as as "homogeneous" Web service community. Moreover, in the user-centric Web environment, the users may want to subscribe these services as RSS/Atom feeds, which is much easier than using UDDI. We studied the secured approach for accessing web services by using SHA-1 algorithm to provide a flexible framework for fulfilling basic security requirements. This security can be enhanced by cryptanalysis of the used cryptographic algorithms so that in future robust algorithms can be proposed for security of the web applications. For the analysis and design of the web application, we can propose the secure web life cycle.

REFERENCES

[1] Xuanzhe Liu, Gang Huang, "Discovering Homogeneous Web Service Community in the User-Centric Web Environment", Member, IEEE, and Hong Mei, Senior Member, , IEEE Transactions On Services Computing, Vol. 2, No. 2, April-June 2009

[2] Nils Agne Nordbotten," XML and Web Services Security Standards", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 11, NO. 3, THIRD QUARTER 2009

[3] L.-J. Zhang, J. Zhang, and H. Cai, Services Computing. Springer and Tsinghua Univ. Press, July 2007.

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity Search for Web Services," Proc. 30th Very Large Database Conf. (VLDB '04), 2004.

[5] C. Petrie, "The Myth of Open Web Services: The Rise of the Service Parks," IEEE Internet Computing, vol. 12, no. 3, pp. 93-95, May/June 2008.

[6] X. Liu, G. Huang, and H. Mei, "Consumer-Centric Service Aggregation: The Method and Framework," J. Software, vol. 18, no. 8, pp. 1883-1895, Aug. 2007.

[7] X. Liu, L. Zhou, G. Huang, and H. Mei, "Consumer-Centric Web Service Discovery and Subscription," Proc. Int'l Conf. e-Business Eng. (ICEBE '07), pp. 543-550, 2007.

[8] X. Liu, L. Zhou, G. Huang, and H. Mei, "Towards a Service Pool Based Approach for QoS-Aware Web Services Discovery and Subscription," Proc. ACM 16th Int'l Conf. World Wide Web (WWW'07), pp. 1253-1254, May 2007.

[9] G. Huang, X. Liu, and H. Mei, "SOAR: Towards Dependable Service-Oriented Architecture via Reflective Middleware," Int'l J. Simulation and Process Modeling, vol. 3, nos. 1/2, pp. 55-65, 2007.

[10] L.-J. Zhang, S. Erickson, and J. Roy, "A Web 2.0 Tune-Up,"IT Professional, vol. 9, no. 3, p. 9, May/June 2007.

[11] E. Al-Masri and Q.H. Mahmou, "Investigating Web Services on the World Wide Web," Proc. 17th World Wide Web Conf., pp. 795-804, Apr. 2008.

[12] B. Benatallah, M.-S. Hacid, A. Leger, C. Rey, and F. Toumani, "On Automating Web Service Discovery," VLDB J., vol 14, no. 11,pp. 84-96, 2004.

[13] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," Proc. Int'l Workshop Semantic Web Services and Web Process Composition, pp. 43-54, 2004.

[14] M. Klusch, B. Fries, and K. Sycara, "Automated Semantic Web Service Discovery with OWLS-MX," Proc. Fifth Int'l Joint Conf. Autonomous Agents and Multi agent Systems, pp. 915-922, 2006.

[15] E.M. Maximilien and M.P. Singh, "A Framework and Ontology for Dynamic Web Services Selection," IEEE Internet Computing, vol. 8, no. 5, pp. 84-93, Sept./Oct. 2004.

[16] M. Zhou, S. Bao, X. Wu, and Y. Yu, "An Unsupervised Model for Exploring Hierarchical Semantics from Social Annotations," Proc .Sixth Int'l Semantic Web Conf., Nov. 2007.

[17] S. Thummalapenta and T. Xie, "PARSE Web: A Programmer Assistant for Reusing Open Source Code on the Web," Proc. 22nd IEEE/ACM Int'l Conf. Automated Software Eng. (ASE '07), pp. 204- 213, Nov. 2007.

[18] A.M. Zaremski and J.M. Wing, "Specification Matching of Software Components," ACM Trans. Software Eng. and Methodology, vol. 6, pp. 333-369, 1997.

[19] Y. Hao and Y. Zhang, "Web Services Discovery Based on Schema Matching," Proc. 13th Australian Computer Science Conf. (ACSC '07), pp. 107-113, Jan./Feb. 2007.

[20] E. Rahm and P.A. Bernstein, "A Survey on Approaches to Automatic Schema Matching," VLDB J., vol. 10, no. 4, pp. 334- 350, 2001.

[21] H.H. Do and E. Rahm, "COMA: A System for Flexible Combination of Schema Matching Approaches," Proc. Very Large Database Conf. (VLDB '02), 2002.

[22] Giovanni Pacifici, Senior Member, IEEE, Mike Spreitzer, Asser N. Tantawi, Senior Member, IEEE, "Performance Management for Cluster-Based Web Services", ieee journal

on selected areas in communications, vol. 23, no. 12, December 2005

[23] E. Bouillet, M. Feblowitz, H. Feng, Z. Liu, A. Rang Nathan, and A. Riabov, "A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition," Proc. IEEE Int'l Conf. Services Computing (SCC '08), pp. 389-396, July 2008.

[24] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms. MIT Press, 2001.

[25] J. Snell, Advertise Web Services with Atom 1.0, http://www-128.ibm.com/developerworks/webservices/library/ws-atomwas/, 2009.

[26] W3C, OWL-S: Semantic Markup for Web Services, http://www.w3. Org/Submission/OWL-S/, Nov. 2004.

AUTHORS PROFILE

Mr. D.R. Ingle (ISTE LM'2004) is Associate Professor of Computer Engineering Department at Bharati Vidyapeeth College of Engineering, NaviMumbai, Maharastra state, India. He received bachelor degree, and Master degree in computer engineering. He has participated in more than 10 refresher courses to meet the needs of current technology. He has contributed more than 20 research papers at national, International Journals. He is life member of Indian Society of Technical Education. His areas of interest are in Databases, intelligent Systems, and Web Engineering.

Dr. B.B.Meshram (CSI LM'95, IE '95) is Professor and head of Computer Technology Department at VJTI, Matunga, Mumbai, Maharastra state, India. He received bachelor degree, Master degree and doctoral degree in computer engineering. He has participated in more than 30 refresher courses to meet the needs of current technology. He has chaired more than 15 AICTE STTP Programs and conferences. He has received the appreciation for lecture at Manchester and Cardip University, UK. He has contributed more than 200 research papers at national, International Journals. He is life member of computer society of India and Institute of Engineers. His current research interests are in Databases, data warehousing, data mining, intelligent Systems, Web Engineering and network security.