Semantic Web Service Clustering Using Concept Lattice: Multi Agent Based Approach

D. Lubin Balasubramanian[#], S.R.Murugaiyan[*], G. Sambasivam[#], T. Vengattaraman[#] and P. Dhavachelvan[#]

[#]Department of Computer Science, Pondicherry University, Puducherry, India.

[*]Research Scholar, Department of Computer Science& Engineering, ManonmaiamSundaranar University, Tamil Nadu, India.

{balu.daya, murugaiyansr, gsambu,vengattaraman.t, dhavachelvan}@gmail.com

***Abstract*—Web Services is the software functionality or the service functionality which was been exposed in the external world by an abstract interface through the network, described using the standard WSDL language, that was been published by the service provider in UDDI business register office, where the requestors can access their required services. Whereas, the web services are nowadays playing a major role in the current scenario and in which the web service discovery has become a real problem, since lack of the public registries to publish and organize the fairly huge number of existing services. The major issue here is to find the relevant web services among the large number of services. Another major issue is to discover the services in less time. One way to address the above issues is to cluster the available services. The existing approaches clusters using lattices based on the operations provided by the web services. The semantic information is not considered in the existing approaches. In this model, we address the issue of semantically clustering the web services using lattices based on multi-agent systems. Lattice based clustering is achieved using Formal Concept Analysis (FCA). A concept lattice is feasible for small to medium size collections. The size of the concept lattice can grow exponentially with respect to the number of context. They cluster web services based on lattices using Formal Concept Analysis (FCA).**

Keyword-Web Services, Clustering, FCA (Formal Concept Analysis), Lattice, Semantics

I. Introduction

A web service can be thought as a web application which uses XML based standards for communicating with external systems for providing the necessary service to the user. The Web Service Architecture working group [5] defines a web service as a software system designed to support interoperable machine-to-machine interaction over a network and has an interface described in a machine-readable format (specifically WSDL) where other systems interact with the system in a manner prescribed by its description using SOAP messages. Web Services are encapsulated, loosely coupled, self-describing, self-advertising, uniquely addressable, standards based and platform independent contracted functions offered through standard protocols [31,4].

Web Service Discovery is the act of locating a machine-process able description of a web service that may have been previously unknown and that meets certain functional criteria. The goal of this research is to discover appropriate web services [5]. A discovery service is a service that facilitates the process of performing discovery of web services from the service registry based on the requirements of the service requestor. It is a logical role, and could be performed by the requester agent, the provider agent or some other agent. The service provider provides the service by registering them at the UDDI (Universal Description, Discovery and Integration) [1,24], which have to be discovered during service discovery. Though progressive and cutting-edge techniques are proposed to address these challenges in conservative environment, still it is in its critical stage at the semantic discovery scenario, particularly when the quality attributes are included. The retrieved web services, at times may not be desirable for the user and the objective of the user desirable service discovery fails.

In this paper, a framework for semantic discovery of services based on the properties clustered using concept lattice is developed based on multi-agent systems [2,3]. A novel similarity measure for assessing the semantic relevance during service discovery is proposed to bridge the semantic gap between the service request and the service provided. A two tier User Preference Model (UPM) is proposed to support the service discovery with respect to the non-functional requests. The tier I of the UPM deals with the qualification of the QoS parameters, where the user is presented with the available quality parameters for defining them in the model. The tier II of the UPM quantifies the qualified QoS parameters, where the user will actually set the preference values.

Concept Lattice based clustering [10] technique is used as lattices provide the inherent relation between the web services that are in the UDDI as the web services are linked using a hierarchical relation (operation in web service discovery). The representation of the web service using concept lattices is relatively simple task as they can be modelled as graph data structure. The cost of time incurred for searching the cluster can be reduced using the proposed framework. The facility of concept lattice technique to include contextual knowledge can be of colossal use in discovering user desirable services.

The contributions of our work can be summarized in the following:

- The use of concept lattice based clustering technique, which bridges the semantic gap between the service requested and the service provided.
- The use of a novel similarity measure, which considers both WordNet [6,25] and Normalized Google Distance (NGD) [8,26] based similarity measure.

The remaining of this paper is organized as follows: In Section 2, we describe the related research work in the area of lattice clustering model. In Section 3, we propose and extend it to support the service selection in Section 4 we further extend our proposed architecture in the form of semantic similarity using the service selection. In Section 5 the similarity metric is been proposed. The implementation aspects of the web service clustering using the lattice model and the analysis of the experimental results is been proved in the Section 6. Finally, the conclusion is given in the Section 7 with future directions.

II. Related Works

Several works have been proposed for web service clustering, in order to facilitate service selection and discovery. The major challenges associated with the discovery of web services are to find an appropriate web service description and to reduce the time taken to discover the web service. These challenges can be addressed by using the semantic information and clustering of web service. There are numerous methods in the literature for clustering web services using the semantic information. A quick overview of some of the works can be obtained from the various papers. Nearest Similarity Score (NSS) [11] proposed a novel approach for web service categorization. They proposed an automatic mechanism which can help service publishers in the categorization task. In their research, a *Nearest Similarity Score (NSS)* which is one of the Measure of Semantic Relatedness (MSR) of each word is calculated for every predefined category. But here the set of words cannot be changed or altered. Only the technical words should be specified so that semantic information will be considered. String Edit Distance[12] proposed a method to cluster web services based on lattices using Formal Concept Analysis (FCA) and multi-agent systems[2,3]. They have used lattice structure for clustering web services and support them with candidate backup services to ensure continuous functionality. Finding the similarity between the service operations are mainly done using the keyword match, Semantic and the service lattice with QoS constraint are not considered.

Logistic Regression [13,28] is mainly proposed to aggregate several matching strategies instead of fixed integration for SWS matchmaking. They exploit the logistic regression model to integrate various matching strategies and to predict the probability of relevance between a query and a service based on their individual matching scores. The similarity measure for matching two names is *Dice's coefficient*. But it is not so effective and appropriate for integrating individual similarity values obtained from various matching strategies on different description components WordNet Based Similarity Measurement method[14**]** improved the service discovery efferent by grouping the similar web services based on the semantic representations are described using 6-tuple and 5-tuple respectively. A WordNet based similarity measurement method for textual elements has been considered for calculating the semantic similarity. The proposed model clusters the services at the cost of increasing pre-processing time. Though the semantic web services are described using various tuples, WordNet mainly used for service description and it is not for the technical word representation for semantic process.The PSO based clustering [15,34] proved that a set of metrics which computes the degree of match between two services. The degree of match is computed based on the concept of hierarchical relations. The evaluation of *degree of match* between two concepts c_i and c_j involves the following steps such as: Exact Match (Equivalence of two different concepts), Fail Match (No relation between the two concepts) and Sibling Match (the two concepts are siblings). The major drawback of this paper is that a lot of computation is involved due to adopting PSO. The outperform classical partitioning techniques as it avoids the problems of local optima stagnation.

OWL-s language concept [16,29,35,36]proposed a method which aims to cluster web services before web service discovery based on OWL-s language. The web service similarity discussed here is based on an accurate concept semantic similarity of the domain ontology. Concept semantic similarity considered. The influencing factors of concept semantic similarity based on the hierarchy is mainly depend on the depth of the hierarchy. The Depth of the hierarchy is been calculated in which the domain ontology hierarchy is said to be deeper and also it shows it is more exhaustive in the concept classification. But here there is no unified algorithm for concept semantic similarity based on domain ontology. Service Clustering [17] proposed to aim at grouping similar services according to the similarity between different services. A framework for semantic service clustering which contains clustering based on profile similarity and process similarity has been modelled. This method can be easily used to discover candidate services as it considers the similarity between the profile and the underlying process. The Service Clustering is mainly for gaining the basic understanding of the cluster but it not suitable for process similarity; hence grouping of the clustered process in the form of lattice the process similarity can be achieved.

Knowledge Discovery in Databases (KDD) [18,30] is the conceptual clustering method, which is well suited for analysing very large databases. Iceberg concept lattices are based on the theory of Formal Concept Analysis (FCA) a mathematical theory with applications in data analysis, information retrieval and knowledge discovery. *String Edit Distance* is used as the similarity measure for identifying similar web services used for clustering. The important work in this research is that concept lattices are used to represent conceptual hierarchies which are inherent in data. Well they are mainly concentrated in the Knowledge Discovery in Databases (KDD) but the semantic approach is missed. By using the word set the semantic approach can be retrieved. Candidate Backup Method [19]is mainly used to discover web services with its candidate substitutes. Concept lattices have been used to classify web services, depending on the similarity estimated between the operations. Lattices are used to discover web services along with candidate backups to ensure the continuous functionality in case of composite web applications. The similarity measure can be used from the research of *Dong, X., et al, Stroulia et al, or Kokash, N., et al,* for identifying structural and semantic service similarity. Though lattices are generated based on their operation similarity, keywords are also used and QoS was not considered.

Supporting Service Selection method [20,33] proposed a concept lattice to support service selection, in their approach based on Formal Concept Analysis (FCA) to understand the relationships between services. This approach allows an analyst to cluster similar services, highlights hierarchical relationships and, in general, commonalities and difference between services. They have developed a tool that provides several service browsing capabilities, and also they have been evaluated with different case studies built upon real sets of services. They showed how these relationships can be derived by lattices built using Formal Concept Analysis (FCA) upon multi-agent and WSDL interfaces. Ontology construction method [21] identified that it is very hard since the domain ontology for each area is not existed. A novel method in the view of user’s requirement for the web service discovery has been proposed. The web services are clustered based on the user’s common requirements. This includes the requirement modelling, service clustering and obtaining service QoS value. RESTful web services [22]has notified that the growing number of RESTful web services available on the web raises a challenging search problem as to how the desired web services should be located. They have proposed a combination method of WADL and a learning ontology mechanism to enable RESTful semantic web services. These syntactic and semantic descriptions allow search engines to support a similarity search for the RESTful web services.

Hence, we suggest considering the various semantic measures used for clustering in the literature for lattice based clustering using multi-agent systems. The semantic similarity measures used by the above researchers are listed below:

- Normalized Google Distance:

$$NGD\ (x,y) = \frac{\max\{log\,f(x)\,,log\,f(y)\}\log f(x,y)}{\log M\ \min\{\log f(x), \log f(y)\}}$$

- Cosine Similarity Measure:

$$Sim_{sd}(sd_i, sd_j) = \frac{\overrightarrow{sd_i}.\overrightarrow{sd_j}}{\left|\left|\overrightarrow{sd_i}\right|\right| * \left|\left|\overrightarrow{sd_j}\right|\right|}$$

- Ontology based Degree of Match Measure:

$$DoM_{c_h}^{sub}(c_i, c_j) = \frac{p(c_i)}{n} \sum_{c_k \in D} DoM_{c_h}^{sub}(c_k, c_j)$$

- Ontology based similarity measure:

$$Sim(C_i, C_j) = Dep(C_i, C_j) * (\alpha Path(C_i, C_j) + \ \beta\rho(C_i, C_j) + \ \delta Anti(C_s, C_t))$$

In summary, it is said that a research on web services clustering in future need to address the above challenges with the attributes mentioned. Thus in our research, we use concept lattices for discovering web services in order to address these issues. We have also proposed a semantic similarity measure for calculating the semantic similarity between the services.

III. BACKGROUND

The primary aim of web service is to facilitate interactions between the software systems using XML standards. The architecture of web services is also designed in such a way that it employs XML standards to define and describe web services. This architecture also enables some of the web services functionalities such as discovery and composition to be done automatically to a certain extent. Fig. 1 shows the classical web services computational model which clearly indicates that there are three major elements such as service provider, service registry and a service requestor with three major operations such as publish, find and bind.

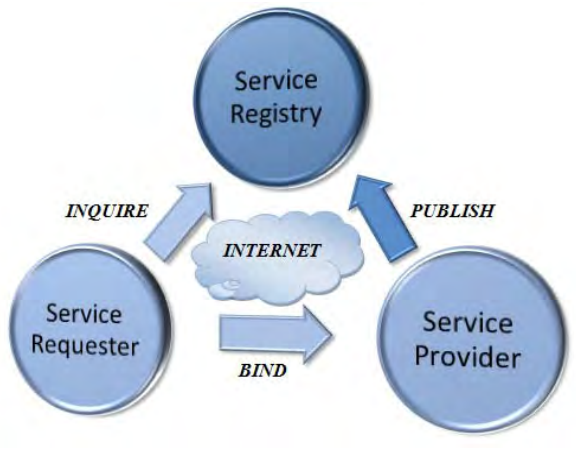


Fig. 1. Web Service Computing Model

The state space representation of the architecture in Fig. 1 can be depicted as shown in Fig. 2.

The basic operations of Web Service Computing are defined as,

$$BOP_{WSC} = \{F, B, P\}$$

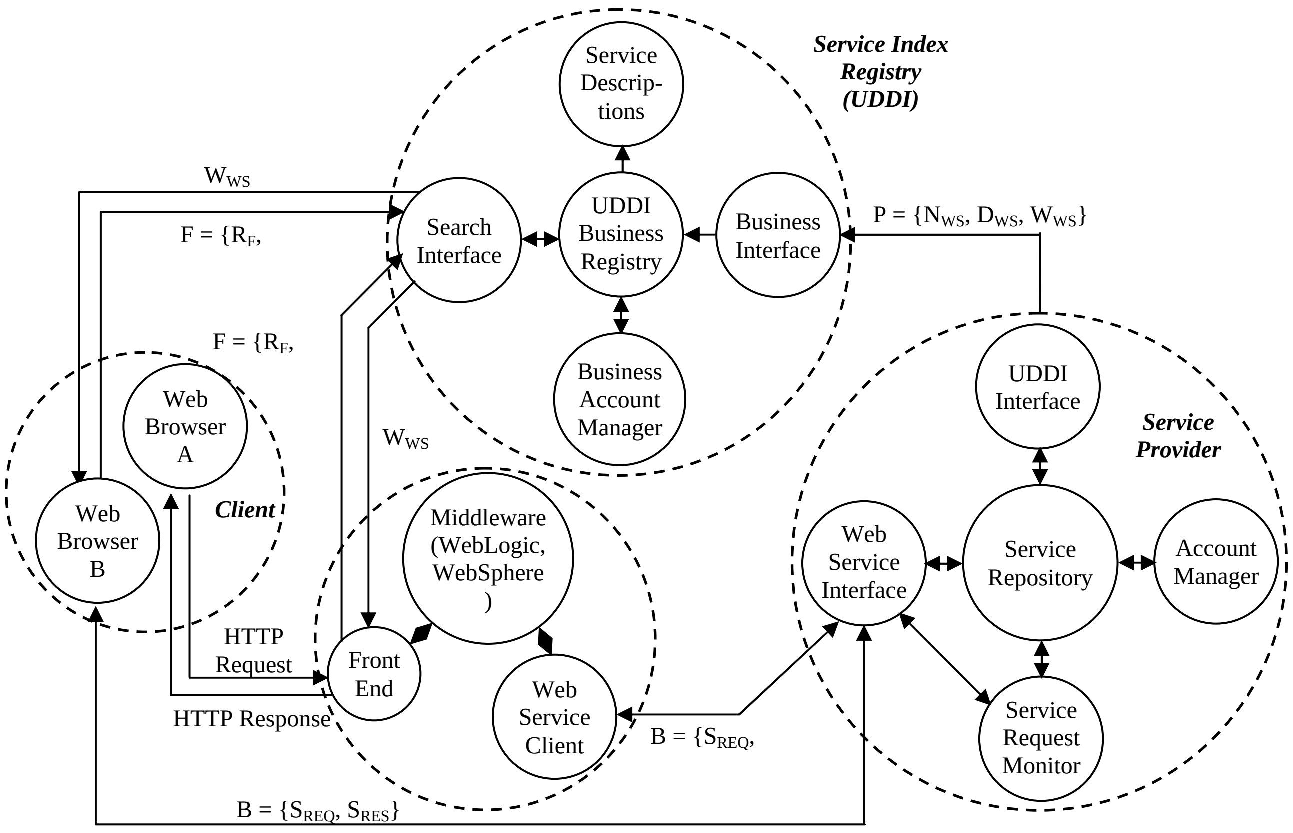


Fig. 2. State Space Representation

where,

- BOP_{WSC} *refers to the Basic Operations of Web Service Computing Model,*
- *F refers to the Find Operation and it can be defined as* $F = \{R_F, R_{NF}\}$*, where*
  - R_F *refers to the Functional Requirements, and*

  - *R_{NF} refers to the Non-Functional Requirements.*
- *B refers to the Bind Operation and it can be defined as B = {S_{REQ}, S_{RES}}, where*
  - *S_{REQ} refers to the SOAP Request, and*
  - *S_{RES} refers to the SOAP Response carrying XML data.*
- *P refers to the Publish Operation and it can be defined as P = {N_{WS}, D_{WS}, W_{WS}}, where*
  - *N_{WS} refers to the Name of the Web Service,*
  - *D_{WS} refers to the Description of the Web Service which is optional, and*
  - *W_{WS} refers to the WSDL description of the web service.*

The *service provider* is the actual owner of the web service from the business point of view. The web service is created by the service provider and it is given a uniform resource identifier which enables the users to use the service. The service provider can also be said as a platform that hosts the web services [32].The *Service Repository* in the service provider is the actual location of the web services. The service repository has a unique internet protocol address using which the S_{REQ} and S_{RES} are made during B. The service repository is accessed by the web services interface in order to satisfy the request and provide response to the service requestor.

The *Account Manager* in the service provider creates an account with the service registry before publishing the web services. The web service descriptions are obtained from the web services in the form of Web Service Description Language (WSDL) files. The *UDDI Interface* component prepares the web service descriptions or models the web service descriptions according to the service registry and *publishes* the service to the service registry. The service provider then tests the registry for the published service and its functional description before making the service available to the public.

The service provider also interfaces with the service requestor with a *web service interface* that binds the service requestor to the service provider. In other words, the service requestor uses the service provided by the service provider using the Web Services Interface module. A *Service Request Monitor* component present in the service provider monitors the usage of its service in the service requestor. The service provider and the service requestor are bound by terms which enable the service requestor to use the service provided by the service provider.

The *service registry* is the central registry of web services where the service providers publish their service which can be searched by the service requestors. This registry is often referred to as Universal Description Discovery and Integration (UDDI). The *Business Account Manager* component manages the accounts that are created by the service providers. This component helps the service providers to create an account before they publish their services. The core functionality of the service registry is to maintain the registry of web services that are being registered with it.

The *UDDI Business Registry* (UBR) is responsible for creating this registry of web services. The service provider publishes the web service using the WSDL file along with the functional descriptions which are parsed by the *Business Interface* component and the service is registered with the UBR with its own description. Each entry in the UBR has its own service description and specifications which the service requestors search for finding the appropriate web services. The *Business Interface* component is responsible for registering the web service description obtained from the UDDI Interface of the Service Provider to the UDDI Business Registry of the Service Registry.

The *Search Interface* component in the service registry is by the service requestors in order to *find* their needed services. The search interface in turn searches the UBR for the requested service descriptions and then returns the WSDL descriptions of the matched service to the service requestor. The clustering in the service registry is performed either by considering the available descriptions during the time of publishing a service or by considering the semantic information of the web service based on the operations provided by the web service.

The *service requestor* may the actual user of the web service or a broker that uses the web service to satisfy the request of its client. The service requestor can be viewed as a business that requires certain functions to be satisfied from business point of view and can be viewed as an application that is looking for initiating an interaction with a service from the application point of view. The service requestor is usually a web browser using which we communicate with the Internet. The service requestor can also be a standalone application or even another web service that requires a service [3].

The service requestor contacts the *search interface* in the service registry to locate the desired web services. A web service client is then created based on the WSDL specifications and the client is made to *bind*with the service provider. A Simple Object Access Protocol (SOAP) request is generated by the client to the *web services interface* in the service provider which contains the data to be requested in the form of XML.

The web service interface then accesses the web service to perform the necessary operation and returns the result in the form of SOAP response carrying XML data. In the case of using a broker to use the web services, then the HTTP request / response is sent to the broker's front end which can be written using HTML, JSP, ASP.NET, etc. The middleware then gets the request from the front end and makes it easy for the web services client to read.

IV. System Design

A. System Architecture

The system is designed in such a way that it address the goals and objectives of the research work described in the earlier discussion. The Fig. 3 depicts the detailed design of the system and the components used in the architecture are described in the following discussion.

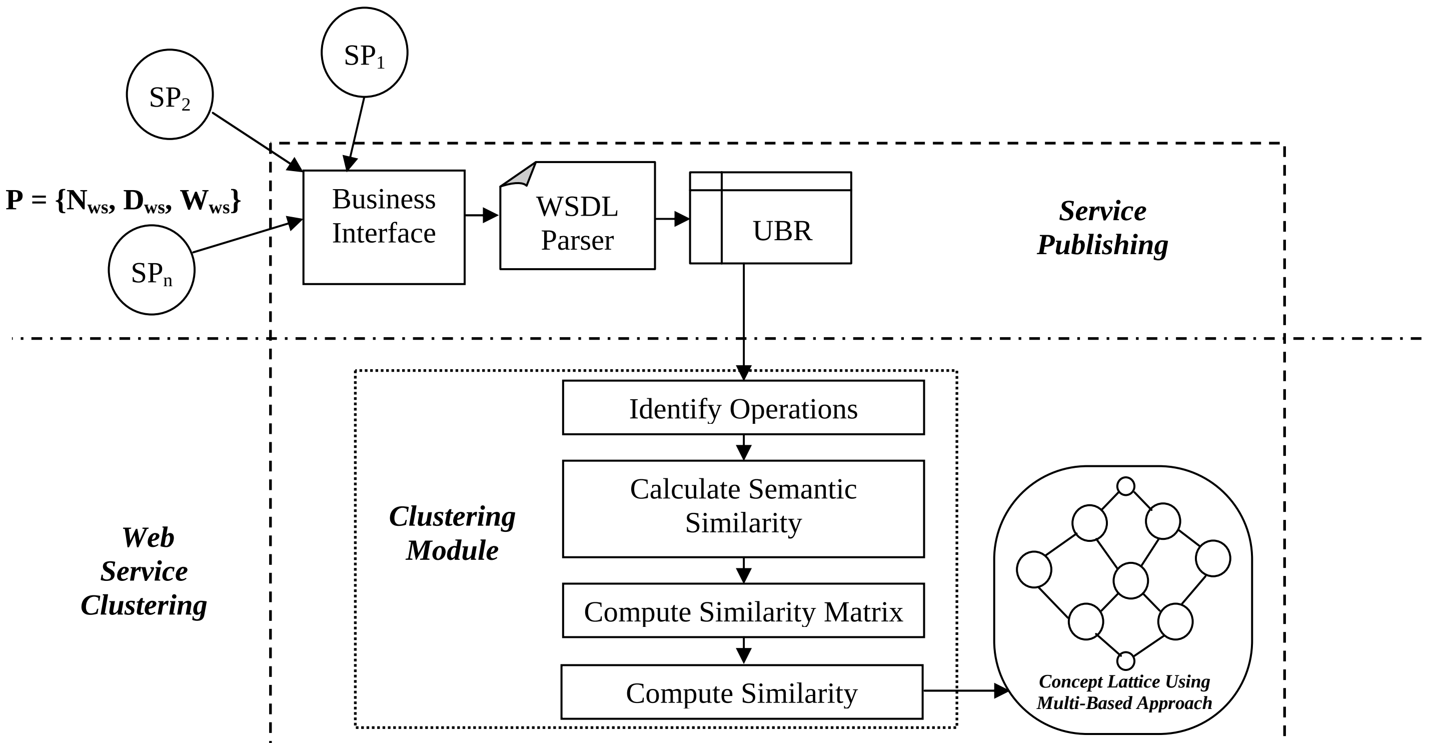


Fig. 3. Generalized System Architecture

The *service provider* is the entity that has the web service and the service repository is maintained by the service provider. The service provider is termed as the actual owner of the web service and he registers the service to the UDDI which can be used by the service requestors. The *business interface* is used by the service providers for registering them as a service provider and to register the available service to the service registry.

Algorithm 1 WSDL Parser

In: Service Descriptions W_{ws}
Out: Service Name N_{ws}, Operations O_{ws} and Location L_{ws}

```
$O_{ws}$ = null
for each service ws in $W_{ws}$
    get Service Name $N_{ws}$
    for each port p in ws
        get Service Location $L_{ws}$
        get PortName, PortType
        for each Operation op in PortType
            $O_{ws} = O_{ws}$ U OperationName
        end
    end
end
```

The business interface receives the service descriptions in the form of a WSDL document and the functionalities are obtained from the service descriptions.

The *WSDL parser* is used to parse the WSDL file for finding the service name, descriptions, operations and end point of the web service. The data obtained from the WSDL parser are tabulated in the UDDI Business Registry (UBR) which is then looked up during the discovery process. The WSDL parser implements the Algorithm 1 presented below which can be viewed as XML parsing.

The *UDDI Business Registry* (UBR) is responsible for creating this registry of web services. The service provider publishes the web service using the WSDL file along with the functional descriptions which are parsed by the *WSDL Parser* component and the service is registered with the UBR with its own description. The *web service interface* is used by the service requestor or the client to search the services available in the service registry. The web service interface is the front end and the client communicates to the service registry using this component only.

Here the Clustering Module clearly explains the process of the formation of the lattice structure using multi-agent systems. In which the registered service by the service provider in the business registry is then parsed by using the parsing algorithm and it finds the operations of the services based on their descriptions and then a matrix form is been generated it is called as SimMat and then based on the this SimCxt is been generated and thus the lattice is been formed.

B. Structure of the lattice node

The Structure of the latticenodes is been depicted in the fig. 4 which contains the syntactic informations and as well as the semantic information with the QoS information.

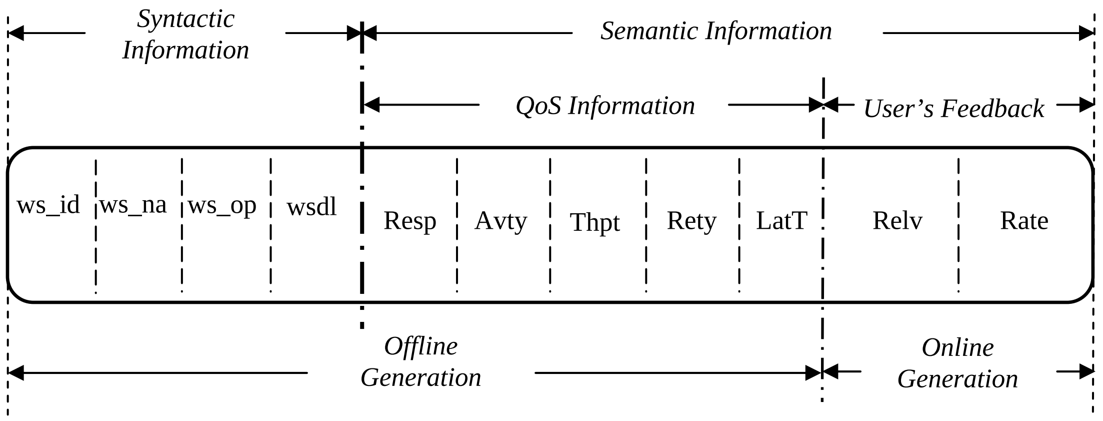


Fig. 4. Structure of the Lattice Node

- *ws_id* is the unique *web service identifier* for easy identification and indexing of web service in the UDDI Business Registry. The search can be easily performed using the numbers as searching and matching a string is more costly than searching numbers.
- *ws_na* is the *web service name* provided by the service provider in which it offers the option to map web service requests based on the qualified name. In terms of the operation the name attribute of the input and output elements provide a unique name among all input and output.
- *ws_op* is the *web service operation* which is provided by the web service and mainly concludes the publication of service descriptions, lookup or finding of service descriptions, and binding or invoking of service based on the service description.
- *wsdl* is the *web services description language* and it provides a machine readable description of how the service can be called, what parameter it expects and what data structure it returns.
- *Resp* is referred to as the *Response Time*, it mainly based on the time duration between a service user sending a request and receiving the corresponding response. It is measured in millisecond.
- *Avty* is referred to as the *Availability*, is the ratio of the number of successful invocations to total invocations. It is measured in percentage.
- *Thpt* is referred to as the *Throughput*, which is the total number of invocations for a given period of time. It is measured in invocations per second.

- *Rety* is referred to as the *Reliability*, which is the ratio of the number of error messages to total messages. It is measured in percentage.
- *LatT* is referred to as the *Latency*, which is the time taken for the server to process a given request. It is measured in milliseconds.
- *Relv* is referred to as the *Relevance*, which is the rank of web service quality and is measured as percentage.
- *Rate* is referred to as the *Rating* is based on the user feedback and is updated every time the user gives rating to the service.

A node in the lattice structure can be defines as a set of fields as,

$$L_n = \begin{cases} (SN, SM) \\ SN = (SN_1, SN_2, \dots SN_n), \quad where, 0 < i \le n. \\ SM = (SMQ, SMU), \\ SMQ = (SMQ_1, SMQ_2, \dots SMQ_k), \quad where, 0 < j \le k. \\ SMU = (SMU_1, SMU_2, \dots SMU_u), \quad where, 0 < k \le u. \end{cases} \quad - \quad (1)$$

Where the following hold:

- L_n is the Lattice node structure,
- SN refers to the Syntactic Information and 'SN_i' is a particular attribute 'i' and 'n' is the number of fields to accommodate the syntactic information and then $0 < i \le n$. Since this node structure offers a *scalar type data structure*, i.e. one field can accommodate only one type of information, 'n' is the number of fields and types of information stored under Syntactic Head.
- SM refers to the Semantic Information and SMQ refers to the QoS information under Semantic Head and SMU refers to the User's Feedback information under Semantic Head.
- 'SMQ_j' is a particular attribute 'j' and 'q' is the number of fields to accommodate the semantic information and then, $0 < j \le q$. Since this node structure offers a *scalar type data structure*, i.e. one field can accommodate only one type of information, 'q' is the number of fields and types of information stored under Semantic QoS Head.
- 'SMU_k' is a particular attribute 'k' and 'u' is the number of fields to accommodate the semantic information and then, $0 < k \le u$. Since this node structure offers a *scalar type data structure*, i.e. one field can accommodate only one type of information, 'u' is the number of fields and types of information stored under Semantic User's Feedback Head.

C. *Concept Lattice*

A formal definition of a concept lattice in accordance with web services can be stated below as defined in [4]:

"A concept lattice defines a hierarchical representation of services and operations, in which a certain concept inherits all the extents (services) of its descendants and all the intents (operations) of its ascendants."

The concept Lattices are mainly used to represent the conceptual hierarchies of services and operation which used to inherent in data using multi-agent systems. It is the fully based on the mathematical foundation theory of Formal Concept Analysis (FCA). We would like to inherit the clustering method in the form of the concept lattices. The extensions of the concepts provide the clusters and the intensions their description. They are mainly called as the UPPER BOUND and LOWER BOUND. Several algorithms are been proposed for computing concept lattices, in which some of the algorithms that are mainly approached line by line for the support of a concept and so it is easily be adapted to compute the iceberg concept lattices and are utilized in the process of Clustering according to the current challenges. In some cases, the concept lattices are been used for clustering the different web services in which the semantic is not been included.

Formal Concept Analysis is the process which mainly offers a formalization of mathematical of the concepts. The service which should be selected from the group of cluster based on the usage of concept lattices to identify relationships between services or between service operations and to help the understanding types defined inside service interfaces. We build concept lattices by applying formal concept analysis (FCA) to the documentation available for the considered set of services.

Formal Concept Analysis is principled way of deriving a concept hierarchy or formal ontology from a collection of objects and their properties. Each concept in the hierarchy represents the set of objects sharing the same values for a certain set of properties and each sub-concept in the hierarchy contains a subset of the objects in the concepts.

In this section, we explain our approach using some basic formal definitions, along with an illustrative example. For clarity sake, we illustrate our approach using an imaginary set of web services for performing calculations. Each service from this set is parsed by a WSDL parser to extract its signatures. The set of services with their signatures are given unique identifiers, as listed in table 1. [10]

Next, a similarity measure must be chosen, and the operations signatures extracted from the WSDL files will be used by this similarity measure, according to its input format. Several similarity measures for web services exist in the literature.

TABLE I
A set of calculation services with their operations

Services	**Id**	**Operations**	**Id**
Mobile	ws_1	GetCost GetModel	op_{11} op_{12}
Sports	ws_2	GetBall	op_{21}
Shipping	ws_3	Cargo GetTicket GetLocation GetPassenger	op_{31} op_{32} op_{33} op_{34}

We evaluate the similarity according to the semantics; the similarity measure is applied on various pairs of operations provided by different services. We do not consider the similarity between operations provided by the same service, because when a service becomes dysfunctional, all of its operations become dysfunctional too. The similarity is assessed in the form of values in the range [0, 1]. If two operations are sufficiently similar, the similarity value will approach 1, or else it will approach 0.

A Similarity measure (*Sim*) can be defined as follows: [12]

$$Sim : O \times O \rightarrow [0,1]$$
$$\forall\, op_{ij} \in O \rightarrow Sim(op_{ij},op_{ij}) = 1$$
(An operation with itself)
$$\forall op_{ij},op_{ik} \in O \rightarrow Sim(op_{ij},op_{ik}) = 0$$
(Operations in the same service)
$$\forall op_{ij},op_{nm} \in O \rightarrow Sim(op_{ij},op_{nm})\ [0,1]$$
(Operations in different services)

The similarity values that are been calculated can be presented by a symmetric square matrix (*SimMat)*, as shown in table 2.

TABLE II
The similarity matrix (SimMat) [12]

	op_{11}	op_{12}	op_{21}	op_{31}	op_{32}	op_{33}	op_{34}
op_{11}	1	0	0.75	0.5	0	0	1
op_{12}	0	1	0	0	0.75	0	0
op_{21}	0.75	0	1	0.75	0	0	0.75
op_{31}	0.5	0	0.75	1	0	0	0
op_{32}	0	0.75	0	0	1	0	0
op_{33}	0	0	0	0	0	1	0
op_{34}	1	0	0.75	0	0	0	1

This matrix is of size $n=|O|$, and its diagonal elements are mostly equal to 1, since we consider that the similarity of an operation with itself is been equal to 1 as said above. From the similarity matrix *SimMat,* we can extract several binary contexts, by specifying threshold values $\Theta \in [0,1]$. Thus, the values of *SimMat* that are greater or equal to the chosen threshold Θ are scaled to 1, while other values are scaled to 0. The binary context that corresponds to $\Theta = 0.75$ is shown in table 3, we call it *SimCxt*. [12]

The *SimCxt* context is a triple *(O,O,RSim*$_\Theta$*)*, where *RSim* is a binary relation indicating whether an operation is similar to another operation or not.

$$(op_{ij},op_{nm}) \in RSim_\Theta \leftrightarrow Sim(op_{ij},op_{nm}) \geq \Theta$$

TABLE III
The Binary context (SimCxt) for ϴ = 0.75

	op_{11}	op_{12}	op_{21}	op_{31}	op_{32}	op_{33}	op_{34}
op_{11}	x		x				x
op_{12}		x			x		
op_{21}	x		x	x			x
op_{31}			x	x			
op_{32}		x			x		
op_{33}						x	
op_{34}	x		x				x

In the resulting operation lattice, groups of mutually similar operations can be identified by the concepts having equal extent and intent sets. We call such concepts as square concepts, because they form square gatherings on the binary context matrix.

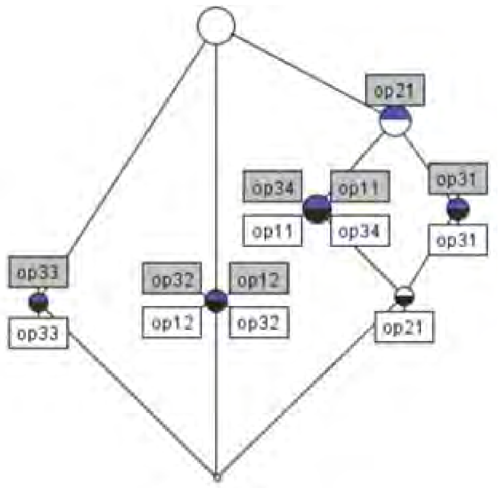


Fig. 5. The Generated Lattice for (SimCxt)

In short, the concept lattices can be used for clustering the web services semantically based on the operations provided by the web services using multi-agent systems. The significance of the concept lattices are mentioned above and the attributes of significance can be seen as:

- Easy to represent
- Linked using a hierarchical relation
- Less time to search
- Can incorporate background knowledge

The mapping model serves as a tool for depicting the importance of concept lattices for clustering of web services semantically with quality factors considered. The figure (Figure 8) below explains the mapping from the attributes of challenges to the attributes of concept lattices.

TABLE IV
No. of Services Vs No .of Domains

Domain Name	Airline	Automobile	Banking	Bio-Informatics	Conversion	Dictionary
No. of Services	7	8	30	223	87	14
Domain Name	Education	Employment	Entertainment	Financial	Library	Messaging
No. of Services	17	19	30	70	16	72
Domain Name	News	Postal	Miscellaneous	Search	Social Networking	Tourist
No. of Services	24	30	149	46	48	18
Domain Name	Tracking	Verification	Weather	***Total No. of Services***		
No. of Services	12	50	30	***1000***		

V. Experimental Setup

The list of real time web service provider stored in our database in our simulation methodology we provide the user with an input text box where user can enter their keyword. Then the entered keyword will be searched for the semantics on the web service description available on the UDDI registry.

This process retrieves a group of related web services for the input keyword. The web service will be parsed the operations and its complex, simple types will be analyzed using Formal Concept Analysis (FCA).

TABLE V
Selected Services and Their Description

Real Time Web Services	**Description**
CDYNE Postal Address Verification	CDYNE Postal Address Verification is a CASS certified API that standardizes, corrects and validates addresses at point of entry or in scheduled batches.
Canada Address Verification	Canada Address Verification Web Service uses the Canada Post national databases, to verify, correct and enhance addresses from any Canadian Location. The service inspects every element of an address
ZIP and Postal Code Information	Instantly retrieve the city, state, country, time zone, latitude and longitude, ZIP and Postal Codes within a radius, and U.S. census information for a given US ZIP or Canadian Postal Code.
Address Doctor Global Address Verification	Verify and correct address in over 240 countries. Plus, it provides additional formatting options like specifying country of origin and preferred language.
StrikeIron US Address Verification	This Web Service verifies and corrects addresses, adds ZIP+4 data, provides delivery point verification, gives congressional Districts, carrier routes, latitude, longitude, and much more.
eCoComa Shipping Rates	Get Shipping rates based off of from and to postal codes for the 4 major United States Shipping services, UPS, USPS, DHL and FedEx.
Address Doctor International Address Quality	The source for International Address Cleansing.
DOTS Address Validation - Canada	Instantly verifies and corrects a Canadian street address. It does this by checking the address against a continuously updated database of Canada Post valid street addresses and other online databases.

The number of parameter, parameter types are analyzed and this gives the web service dependency. We use Google API to compute the Normalized Google Distance (NGD).The WordNet and Normalized Google Distance together considered as the combined score would be more appropriate for measuring the relatedness of the web service operations. Two similarity measures are chosen since, they measure the semantic similarity between the words as assessed by Google Search Engine and WordNet.

In the current situation there is no standard web service testing so the experimental setup is difficult. As analyzed from the previous work, various similarity algorithms are been tested. Here it is to be proved by keeping 1000 Web Services which provide different operations, and these services are been segregated by 20-30 domains. In which some of the services are not in progress currently. Though, it is been added for the purpose if any of the service gets break while clustering it should take an alternative path to show the next priority of related service.

The above table shows the various web services and their corresponding descriptions. The Formal Concept Analysis (FCA) and concept lattices are mainly used to highlight the relationships holding between the service and their operations. Concept lattices are mainly built or obtained from keyword extraction of services. The keywords are extracted from service description and the operation of the services; they are used to support the understanding of the web services through their interfaces.

A. Identifying the Keywords from various services

The main process of this paper is mainly suits to the keyword extraction. The WSDL files are the interface which was been preprocessed in order to extract the keywords from the given service description by representing the attributes. Whereas the lattice is then built, for this process words are been extracted from service WSDL, performing semantic analysis and indexing the words and identifying key words by use of the WordNet.

The extracted words are then preprocessed. Composite words may be split, this may be the case of the operation and parameter names (e.g., “Verification and Validation”)

B. Word Indexing and Keyword Identification

Each element of the vector corresponds to a word or a term in a vocabulary extracted from the service. If $|v|$ is the size of the vocabulary, i.e., the number of different words extracted from the examined description.

$$tf_{i,j} = \frac{Number\ of\ occurences\ of\ word\ j-th}{Total\ Number\ of\ words\ in\ the\ Document\ D_i}$$

According to the above formula or metric the *j-th* element $d_{i,j}$ is derived from the term frequency $tf_{i,j}$ of the *j-th* element in the description D_i and the inverse document frequency idf_j of the term over the entire set of documents. The term frequency $tf_{i,j}$ is defined as above. And the inverse of the document frequency idf_j is defined as: [20]

$$idf_j = \frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ the\ word\ j-th}$$

And the vector element $d_{i,j}$ is

$$d_{i,j} = tf_{i,j}.\log(idf_j)$$

Once the words have been filtered out (and thus the keywords been identified), the context can be identified as the inclusion relationship of keywords into descriptions.

A similarity measure is proposed that calculates the semantic similarity of the operations provided by the web services. The proposed similarity measure considers the harmonic mean of WordNet and Normalized Google Distance for calculating the similarity between the operations. If op_{ij} and $op_{kj}$are the two j^{th} operation provided by the two distinct web services ws_i and ws_k, then the semantic similarity can be calculated as ***SemSim(op$_{ij}$, op$_{kj}$)***,

$$SemSim(op_{ij}, op_{kj}) = \frac{2 * WSim(op_{ij}, op_{kj}) * NGD(op_{ij}, op_{kj})}{WSim(op_{ij}, op_{kj}) + NGD(op_{ij}, op_{kj})}$$

Where,

- WSim(x,y) is the WordNet similarity score of the two words x and y.
- NGD(x,y) is the Normalized Google Distance between the words x and y.

C. Tool Support

Here according to the proposed approach we have built a tool, Concept Explorer (ConExp) which allows the user or the client browsing the services using the concept lattice using multi-agent systems. Given a set of related services, the tool analyzes their relationship or the interfaces (as said above) and represents them as a lattice. The tool provides several querying and browsing features. [20] The tool has been implemented in JAVA. The tool implements the FCA algorithm and graphical features; it is able to draw concept lattices starting from a context table edited by the user. It allows the following

- WSDL parsing capabilities
- Automatic extraction of words from service description
- Querying, Browsing capabilities

VI. Result Analysis

Validating and justifying the performance of the proposed lattice based web service clustering approach.

The extracted words are then preprocessed. Composite words may be split; this may be the case of the operation and parameter names (e.g., "Verification and Validation")

Successive words are been filtered by means of the keywords, stop-list and the normalized Google Distance. According to this the SimCxt is been generated for a particular service as shown in Table VI.

In this context, Wordnet relationship can be exploited to relate the synonyms sets to a single attribute. Hence by using this readability and the usability will be improved in the lattice based on multi-agents using the synonyms sets which is used to check whether different service descriptions actually refer to the same concept in different words. It also offers an extensive ontology that can be used to define entire classes of concepts.

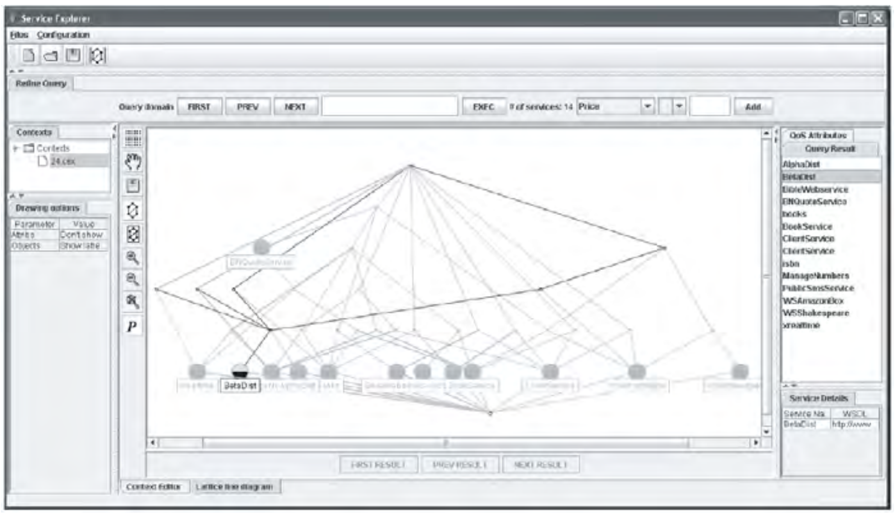
TABLE VI
SimCxt for Selected Services and Their Operations

	VerifyAddressCanada	Verify Address	GetCityNamesForZipCode	GetUrbanizationListForZipCode	GetZipCodesForCityAndState	GetCongressionalDistrictByZip	GetZipCodesWithinDistance
UserID	x						
Password	x						
AddressLine1	x	x					
AddressLine2	x	x					
StreetNumber		x					
Firm							
PreDirection							
StreetName		x					
StreetType							
PostDirection							
Extension							
ExtensionNumber							
Village					x		
City		x		x	x		
AddressStatus							
ZipCode	x		x	x		x	x

In this evaluation, the validation of various process like Clustering, Semantic Clustering, and Lattice Clustering Semantically. Here we have a list of real time web service provider stored in our database in our simulation methodology we provide the user with an input text box where user can enter their keyword. Then the entered keyword will be searched for the semantics on the web service description available on the UDDI registry. This process retrieves a group of related web services for the input keyword. The web service will be parsed the operations and its complex, simple types will be analyzed using Formal Concept Analysis (FCA).

In depth we are analyzing the no of parameter, parameter type and this gives the web service dependency. We use Google API to compute the Normalized Google Distance (NGD).Not only this, to prove that when compared to the other methods lattice has the high performance. For that, we have done the normal clustering Technique K-means. K-means was noticed as one of the popular clustering technique. So, it is taken into the account and it is been verified.

Next, in the clustering the Semantic information is included and then it is been clustered. It is also taken into the account to show the performance of the semantic lattice clustering. The Parameters like Response Time, Availability, Reliability, through put and Latency Time are been taken into the account separately. As it is been validated it is known that the response time is been compared with the workload.



Parameter	Value
Attribs	Don't show
Objects	Show label

AlphaDist
BetaDist
BibleWebservice
BNQuoteService
books
BookService
ClientService
ClientService
isbn
ManageNumbers
PublicSensService
WSAmazonBox
WSShakespeare
xmltime

Service Na	WSDL
BetaDist	http://www

Fig. 6. The Generated Lattice for applied services

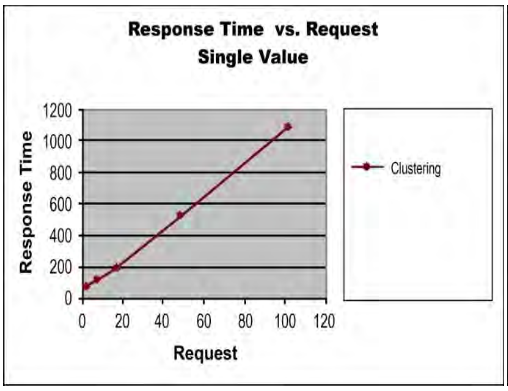


Fig. 7(d). Number Of Requests Vs Response Time For Normal Clustering method

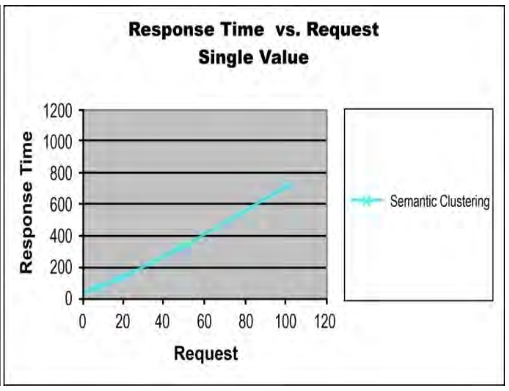


Fig. 7(e). Number Of Requests Vs Response Time For Semantic Clustering method

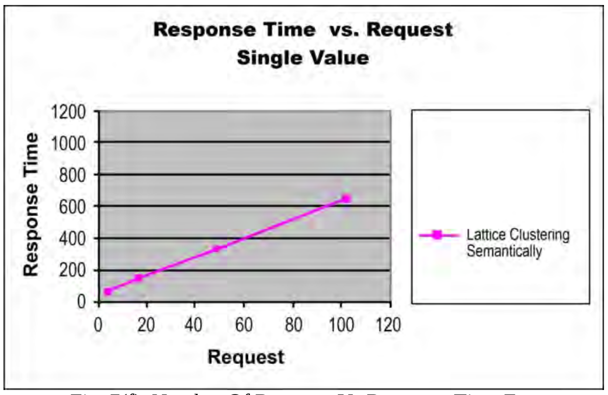


Fig. 7(f). Number Of Requests Vs Response Time For Semantic Lattice Clustering method

Then the Response time is been compared with the number of request by the client or the user. As, it is compared semantically with full scan clustering, semantic clusters and the lattice clustering using multi-agents methods. In which the Lattice allows the highest performance when compared to the previous works. The Response time compared with the workload and the number of requests is been compared together. Hence, it is notified that the Lattice Based Clustering using multi-agent systems allows high performance semantically.

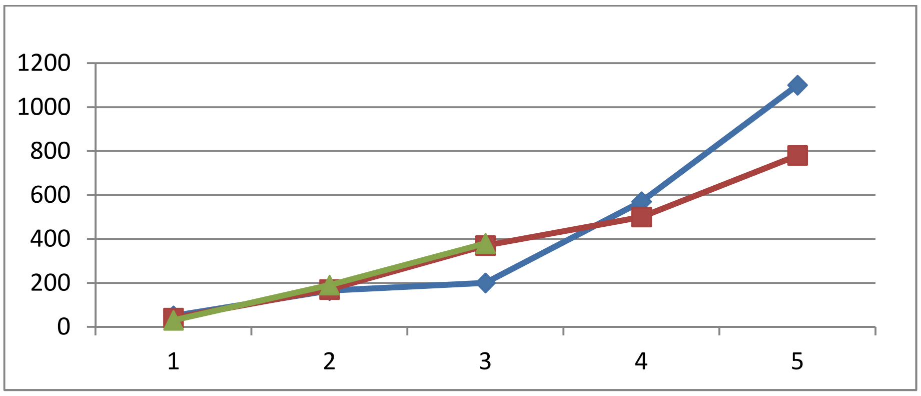


Fig. 7(g). Comparison of Workload Vs Response Time for All the Three Methods

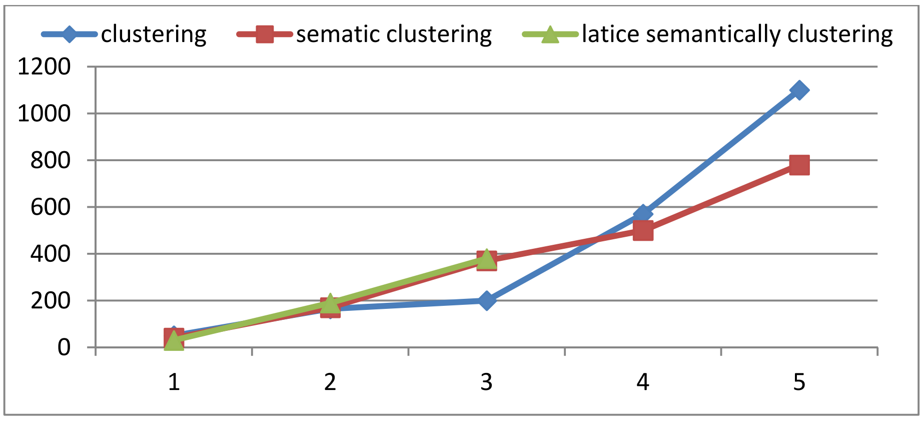


Fig. 7(h). Comparison of Requests Vs Response Time for All the Three Methods

VII. Conclusion and Work-In-Progress

This paper presented an approach to support the service understanding purpose. Actually the basic service discovery mechanism didn't provide any information about the relationships existing between the registered services but it works under the process of list of services relevant to a query. We have been showed that how the relationships can be retrieved by multi-agent based lattice built using Formal Concept Analysis upon the WSDL files descriptions. In this the lattice allows us to understand the similarity between the services relationship. Whereas, the ConExp supports the service understandings and provides browsing and querying features. Hence the lattice is built upon the Keyword extraction from the service description and from the operation parameters. This is been achieved by using the Normalized Google Distance and WordNet.

Future work will be fully dedicated to improve the proposed technique by allowing the user to discover the services by using the given services selected from the lattice through the UDDI registries and from other similar services.

Acknowledgment

This work is a part of the Research Projects sponsored under the Major Project Scheme, UGC, India, Reference Nos: F. No. 41-616/2012 (SR), dated 18 July 2012 and F. No. 40-258/2011(SR), dated 29 June 2011. The authors would like to express their thanks for the financial supports offered by the Sponsored Agency.

References

[1] T. Vengattaraman, S. Abiramy, P. Dhavachelvan and R. Baskaran (2011), "An Application Perspective Evaluation of Multi-Agent System in Versatile Environments", International Journal on Expert Systems with Applications, Elsevier, Vol. 38, No. 3, pp. 1405-1416.

[2] Platon, E., Sabouret, N., & Honiden, S. (2008), "An architecture for exception management in multi-agent systems", International Journal of Agent-Oriented Software Engineering Archive, 2(3), 267–289.

[3] Kaminka, Gal A. (2009) "Detecting disagreements in large-scale multi-agent teams", Autonomous Agents and Multi-Agent Systems archive, 18(3), 501–525.

[4] Guruge Anura, "Web Services Theory and Practices", ELSEVIER Digital Press, ISBN 1-55558-282-6, 2004.

[5] W3C Web Services Architecture Working Group, "Web Service Architecture", Available at http://www.w3.org/TR/ws-arch/

[6] P. Dhavachelvan and G.V.Uma (2003), "Multi-agent Framework Construction for Intra Class Testing of Object-Oriented Software", 18th ISCIS 2003, Springer Verlag - Lecture Notes in Computer Science (LNCS), Vol. 2869, pp. 992-999. ISSN: 0302-9743.

[7] Christensen Erik, Curbera Francisco, Meredith Greg, Weerawarana Sanjiva, "Web Services Description Language (WSDL)", http://www.w3.org/TR/wsdl, W3C Note 15 March 2001.

[8] Paul P.V., T. Vengattaraman, P. Dhavachelvan (2010), "Improving efficiency of Peer Network Applications by formulating Distributed Spanning Tree", Proceedings - 3rd International Conference on Emerging Trends in Engineering and Technology, ICETET 2010, Art. no. 5698439, pp. 813-818 .

[9] Sihem Amer-Yahia, Yannis Kotidis, "Web-Services Architecture for Efficient XML Data Exchange", Proceedings of the 20th International Conference on Data Engineering, IEEE CONFERENCE, 2004.

[10] P. Dhavachelvan, G.V. Uma and V.S.K.Venkatachalapathy (2006), "A New Approach in Development of Distributed Framework for Automated Software Testing Using Agents", International Journal on Knowledge –Based Systems, Elsevier, Vol. 19, No. 4, pp. 235-247, August-2006.

[11] Batra Shalini, Bawa Seema, "Web Service Categorization Using Normalized Similarity Score", International Journal of Computer Theory and Engineering, Vol. 2, No. 1, February 2010, 1793-8201.

[12] Azmeh Zenia, Huchard Marianne, Messai Nizar, Tibermacine Chouki, "Using Concept Lattices to Support Web Services Composition with Backup Services", Publications du Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, lirmm-00533074, version 1 -5 Nov 2010.

[13] P. Dhavachelvan and G.V.Uma (2005), "Multi-agent based Framework for Intra-Class Testing of Object-Oriented Software", International Journal on Applied Soft Computing, Elsevier, Vol-5, No.2, pp. 205-222.

[14] Wen Tao, Sheng Guojun, Li Yingqui, Guo Quan, "Research on Web Service Discovery with Semantics and Clustering", 978-1-4244-8625-0/11, 2011 IEEE.

[15] D.Chandramohan,T.Vengattaraman,D.Rajaguru,R.Baskaran, P.Dhavachelvan, "A Privacy Preserving Representation for Web Service Communicators' in the Cloud", 9th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, India, QSHINE-Proceeding Springer LNICST-Series ISBN: 978-3-642-37948-2, Vol.115, pp.496-506(2013).
[16] P. Victer Paul, D. Rajaguru, N. Saravanan, R. Baskaran and P. Dhavachelvan, "Efficient service cache management in mobile P2P networks", Future Generation Computer Systems, Elsevier, Volume 29, Issue 6, August 2013, Pages 1505–1521.
[17] Sun Ping, "Service Clustering based on Profile and Process Similarity" Third International Symposium on Information Science and Engineering, 978-0-7695-4360-4/10 IEEE-2010.
[18] S. Venkatesan, P. Dhavachelvan and C. Chellapan (2005), "Performance analysis of mobile agent failure recovery in e-service applications", International Journal of Computer Standards and Interfaces, Elsevier, Vol-32, No.1-2, pp. 38-43. ISSN:0920-5489.
[19] Azmeh Zeina, Hamoui Fady, Huchard Marianne, Urtado Christelle, "Backing Composite Web Services Using Formal Concept Analysis" International Symposium LIM – 2010.
[20] Aversano Lerina, Bruno Marcello, Canfora Gerardo, Di Penta Massimiliano, Distante Damiano, "Using Concept Lattices to Support Service Selection", International Journal of Web Services Research, Oct-2006.
[21] Xu Lei, Xu Baowen, Chen Lianjie, Yang Hongji, "Web Services Discovery Based on User Requirements", IEEE International Conference on High Performance Computing and Communication, 978-0-7695-4538-7/11, 2011 IEEE.
[22] Lee Yong-Ju, Kim Chang-Su, "A Learning Ontology Method for RESTful Semantic Web Services", International Conference on Web Services – IEEE 2011.
[23] Gottschalk K, Graham S, Kreger H, Snell J, "Introduction to Web Services Architecture", IBM Systems Journal, Vol. 41, No. 2, 2002.
[24] "Introduction to UDDI: Important Features and Functional Concepts", http://www.oasis-open.org, October 2004.
[25] Box Don, Ehnebuske David, Kakivaya Gopal, Layman Andrew, Mendelsohn Noah, Frystyk Nielsen Henrik, Thatte Satish, Winer Dave, "Simple Object Access Protocol (SOAP) 1.1", http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, W3C Note 08 May 2000.
[26] Ouzzani, M.; Bouguettaya, A., "Efficient access to Web Service", IEEE Journal, DOI: 10.1109/MIC.2004.1273484, Volume 8, Issue 2, 2004, pp.34-44.
[27] Azmeh Zenia, Huchard Marianne, Messai Nizar, Tibermacine Chouki, "Many-Valued Concept Lattices for backing Composite Web Services", Publications du Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier, lirmm-00557258, version 1 - 18 Jan 2011.
[28] DengPing WEI, Ting WANG & Ji WANG, "A Logistic Regression Model for Semantic Web Service Matchmaking", SCIENCE CHINA Information Sciences, Vol. 55 No. 7: 1715-1720 July 2012.
[29] XIE Ling-li, CHEN Fu-zan, KOU Ji-song, "Ontology-Based Semantic Web Service Clustering", 978-1-61284-449-7/11 IEEE-2011.
[30] Stumme Gerd, Taouil Rafik, Bastide Yves, Lakhal Lotfi, "Conceptual Clustering with Iceberg Concept Lattices", International Symposium LIM, CNRS FRE2246-2010.
[31] Ouzzani, M.; Bouguettaya, A., "Efficient access to Web Service", IEEE Journal, DOI: 10.1109/MIC.2004.1273484, Volume 8, Issue 2, 2004, pp.34-44.
[32] P. Victer Paul, N. Saravanan, S.K.V. Jayakumar, P. Dhavachelvan and R. Baskaran (2012), "QoS enhancements for global replication management in peer to peer networks", Future Generation Computer Systems 28 (3), pp. 573-582.
[33] D. Chandramohan, T. Vengattaraman, P. Dhavachelvan, R. Baskaran, V.S.K.Venkatachalapathy, "FEWSS-Framework to Evaluate the Service Suitability and Privacy in a Distributed Web Service Environment", Int. J. Model. Simul. Sci. Comput, 2013. DOI: 10.1142/S1793962313500165.
[34] Nagy Aliz, Oprisa Ciprian, Salomie Ioan, Bianca Pop Cristina, Rozina Chifu Viorica, Dinsoreanu Mihaela, "Particle Swarm Optimization for Clustering Semantic Web Services", 978-0-7695-4540-0/11 IEEE-2011.
[35] Uma.R, Krishnappriya, Dhavachelvan, P, "New Modified Elmore Delay Model For Resistance-Capacitance-Conductance (RCG) Interconnect Network Scheme" Journal of Theoretical and Applied Information Technology, Volume 54, Issue 3, August 2013, Pages 361-371.
[36] Uma. R, Dhavachelvan. P, "Performance enhancement for variable block optimization in FGPA synthesis process" 2nd International Conference on Computational Science, Engineering and Information, CCSEIT 2012, ACM International Conference Proceeding Series, Pages 237-243.