IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Volume-4, Issue-9 E-ISSN: 2347-2693

Web Resources Development Methdology Based on Web Composition Using Ontology for User's Optimal Goal

G.Narayanan[1*] and Pon Periasamy[2]

[1*]PG and Department of Computer Science, Nehru Memorial College, Trichy
[2]PG and Department of Computer Science, Nehru Memorial College, Trichy

Available online at: www.ijcseonline.org

Received: 21/Aug/2016 Revised: 01/Sept/2016 Accepted: 15/Sept/2016 Published: 30/Sep/2016

Abstract— The proposed algorithm expands the meaning of a user's goal using ontology then derives a group of keywords to discover services and web composition are used to select the web services based on QoS to find the optimality solution of their user goal. The efficiency of the web service matching and composition becomes more important than ever because of the vast number of the web services. For this purpose we propose a web service composition algorithm based on the annotated ontology using semantic matching to achieve exact service for user's constraint. We design a resource graph to represent the semantic relationship among Web resources. By analyzing the relations among Web resources and using ontologies, A semantic web services would require careful usage combined technologies this semantic web service is realized to show that they ensure interoperability. Four aspects of web services are presented 1) Standard of XML web services 2) Semantic annotation 3) Web service composition 4) Performance Evolution. Our framework can generate ad-hoc processes for composing Web resources. We have built a prototype to demonstrate that the repetitive tasks in the Web resources can be automatically and tracked and the user can change simple Web resources into reusable services by annotating the data with them.

Keywords- *Ontology, Service-oriented architecture, Service composition, Service discovery and Web services*

1. INTRODUCTION

The word "ontology" was widespread quoted in the Artificial Intelligence domain in recent years. A lot of definitions about ontology are being proposed constantly. Most often quoted definition is that Gruber proposed in 1993. "Ontology is a formal, explicit specification of a shared conceptualization". In definition the "conceptualization" is the abstract model of the phenomenon in existence, the word "shared" points out the ontology is shared and belonged to the collective not individual. The meaning of formal is machine can read and understood the ontology. Ontology's contribute to resolve semantic heterogeneity by providing a shared comprehension of a given domain of interest. Furthermore, the main challenge of interoperability and data integration is still ontologies matching. The work in semantic Web demonstrates how ontologies can be used to address interoperability problems at the application level. Specifically, ontologies have been used during discovery to express the capabilities services, as well as the requests for capabilities. Ontologies are used to improve communication between any user by Specifying the semantics of the symbolic apparatus used in the communication process. More specifically, Jasper and Uschold (1999) identified three major uses of ontologies: (i) to assist in communication between human beings, (ii) to achieve interoperability among software systems, and (iii) to improve the design and the quality of software systems. The clear definition of logic rules will let ontology has stronger functions. The computer will understand the meaning of web pages through linking concepts to concepts under the ontology proposes a method based on the annotated ontology to fulfill the semantic match, but not to use it for composition. We not only annotate ontology for semantic match but for the web service composition. To annotate the ontology, we add two map type variables "in" and "out" to ontology to store the annotation. In the "in" set, the concept and the service id which has the concept as input parameter is stored, in the "out" set, the concept and the service id which has the concept as output parameter is stored, their initial states are empty.

1.1 Web Composition Method

A method to compose a serial of web service to satisfy a query based on the annotated ontology. When a service is registered, it will be mapped to the concepts of the ontology. When a query comes, we can get the corresponding web services quickly by its input or output concepts, not need to traverse all the web services in the service registry. The method needs to preprocess the ontology[3] ,which can cost some time and space ,but the time is only once when web service is registered .With it we can save much time in dynamic compose web services for any query.

1.2 Dynamic Composition Method:

Web service dynamic composition based on an annotated ontology. The capability parameters of the registered web services are used to annotate the domain ontology, when a request comes, only the related web service according to the annotated ontology will be matched or be composed[14]. With the method, the efficiency can be improved significantly when a large number of web services exist.

1.4 Semantic Web Community:

The semantic Web community's[21] responses to the interoperability problem are based on the principles of reasoning about ontologies and understanding how different systems can work togather. The work in semantic web services demonstrates how ontologies can be used to address interoperability problems at the application level. Specifically, ontologies have been used during discovery to experess the capabilites of servicves, as well as the requests for capabilities. Semantic web services seem to be a good choice for loosely coupled architectures. Its success and its popularity are mainly due on one hand to SOA and SOAP protocols and on the other hand to semantic annotations as follows:

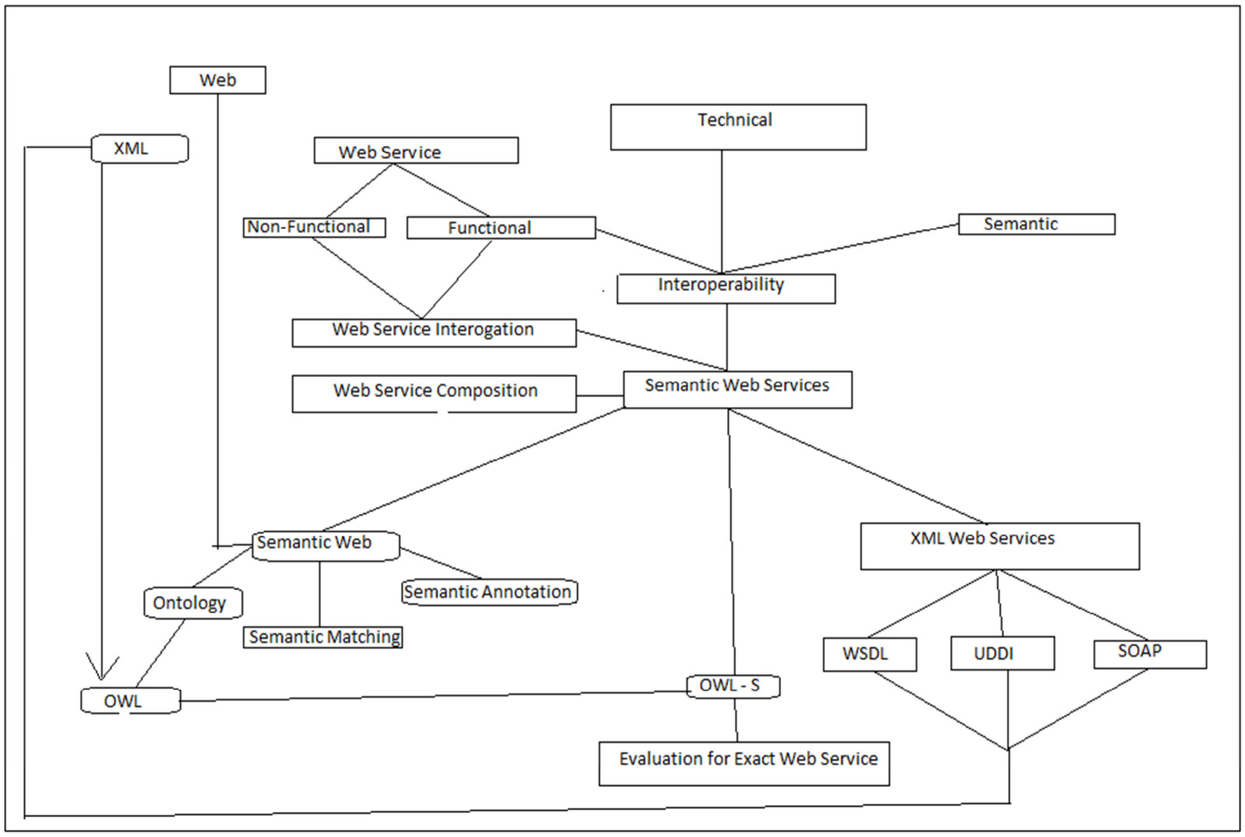


Figure.1 Semantic Web services cartography

1.3 Semantic Annotation

The semantic web services are at the convergence of two signification fields of reaches which are technologies of the internet and XML web services. The purpose of semantic web services is to create a semantic web of services whose properties, interfaces and effects are described in a non-ambiguous and exploitable way by software agents. An annotation assigns to an entity, which is in the text, a link to its semantic description. A semantic annotation is referred to ontology. The idea is to have data through the web defined and linked in such a way that its meaning is explicitly interpretable by software processes rather than just being implicitly interpretable by humans [5]. Semantic annotation can be applied to any web resources. The semantic annotation as follows:

- ❖ SOA (Functional Interoperability): The SOA principles are realized by web services standards and technologies based on XML.
- ❖ SOAP protocol (Technical Interoperability): Web protocols are usually allowed through a firewall and the associated computational cost may be relatively low, due to the possibility of selective of selective encryption and/or signature of SOAP messages. By using SOAP different applications can read and send messages over HTTP to each other.

- Semantic annotation (operational interoperability): It facilitates semantic interoperability of data because they refer to ontologies' describing. Semantically described, services will enable better service discovery and allow easier interoperation and composition. Research in semantic web has shown that annotation with meta-data can help us to solve the problem of inefficient keyword based search in the current web. The concept of annotation can be extended to web services to envision semantic web services[20].

2. AN ONTOLOGY DEFINITION MODEL

An ontology expresses common entities (e.g., people, travel, and weather), and the relations among those entities. Ontology can be visualized as a graph that contains nodes representing entities and edges representing relations among the entities. . The entity "Travel" is related to four more specific entities: "Transportation", "Accommodation", "Tourist Attraction" and "Car Rental".

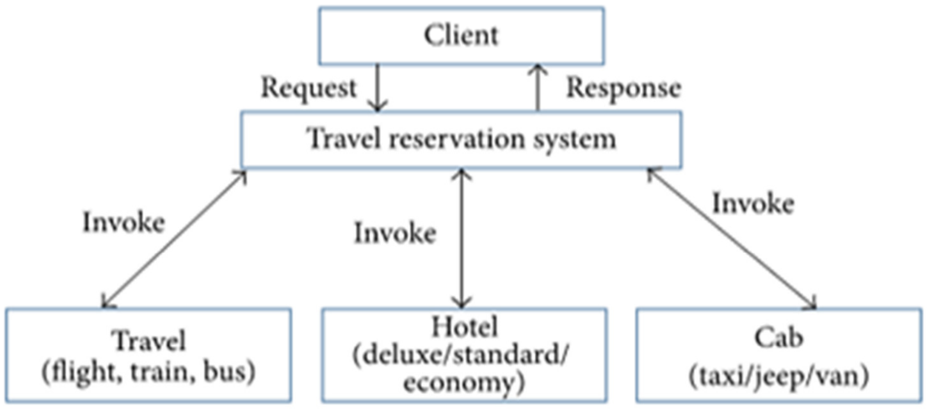

Client
Request
Response
Travel reservation system
Invoke
Invoke
Invoke
Travel (flight, train, bus)
Hotel (deluxe/standard/ economy)
Cab (taxi/jeep/van)

Figure.2 An example of ontology for defining the entity "Travel"

For example, the goal for planning a trip can be expressed using keywords, such as "Trip" or "Travel". To derive the tasks that achieve the specified goal, we analyze the semantic meaning of the specified goal using ontologies. Ontologies capture the information related to particular goals using expert knowledge. For example the ontology for the concept "Travel" lists relevant concepts, such as "Flight", "Hotel Reservation", and "Tourist Attraction". To have a better understanding of the specified goal, we search for existing ontologies that can expand the meaning of a specified goal .Furthermore, we provide an algorithm that analyzes the identified ontology to dynamically discover services and compose an ad-hoc process to achieve the specified goal. We take an ontology which matches with the goal description as the input. The algorithm uses a stepwise approach to discover and organize the Web resources according to the level of abstraction[12]. The high level entities in an ontology graph convey more abstract meanings suitable for discovering Web resources offering general purpose services. Such services allow users to receive the desired Web resources.

2.1. REPRESENTING A RESOURCE GRAPH

We create a resource graph to represent Web resources and the relations between Web resources. We consider the resource graph as a semantic network model which consists of entities and relationships. Entities in a resource graph denote Web resources identifiable using URIs. A Web resource may be linked to other Web resources by a set of relations. We have identified three types of relations:

Data flow based relations: Data flow relations define the flow of data between two or more resources. The data flow relation is determined by matching the schema between the input and output of methods in Web resources. We use link specification [3] to describe the data flow based relation between Web resources in a resource graph.

Transitions based relations:
The response of a Web resource contains next state transition information. A user agent can decide next state based on the semantics of the relations defined in the links available in the response. The relations are used to recommend new Web resources, identify similar Web resources, and define the relationship between the Web resources. Similar to data flow relations, we use link specification to describe the transition based relations[7].

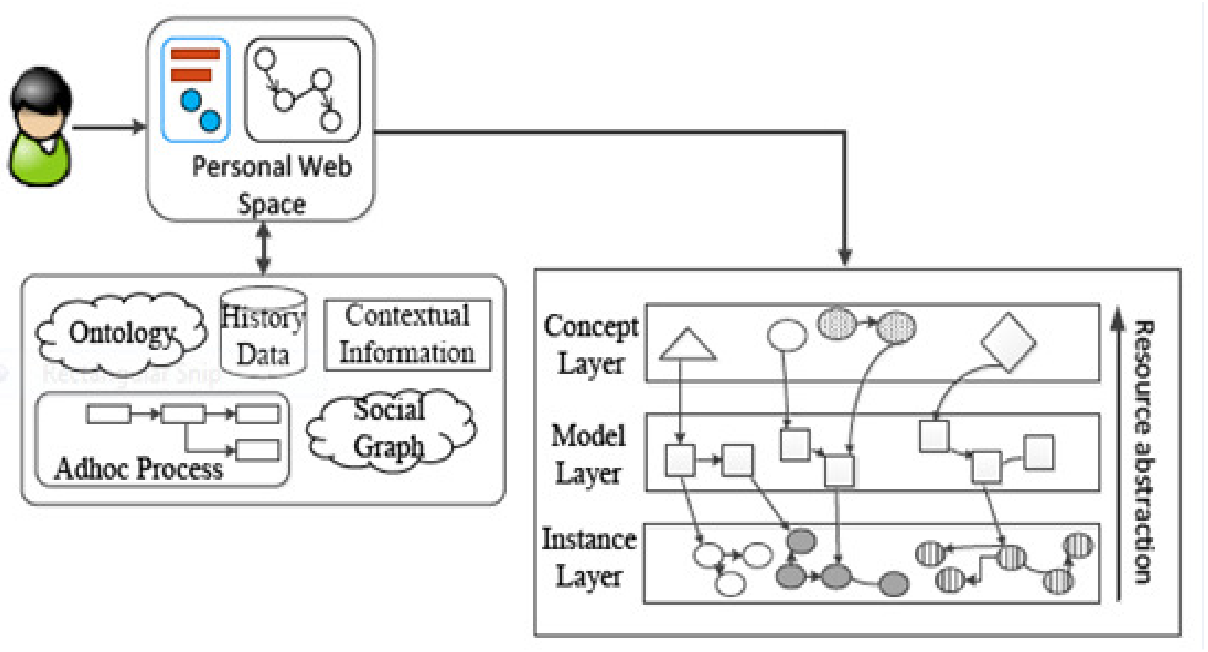


Figure.3 Architecture for personal web space

3. Exiting work and its algorithm

A query for "Travel from source to destination ," Web services for "Travel" may only provide limited services, such as booking flight and train tickets and renting car,. In this case, the end-users may need more specialized services for transportation, accommodation, and tourist attractions. To offer end-users with more options the keywords obtained are travel, mode-of-transport, taxi, hotel, tourist-interest, and so forth. The keyword "travel" derives from the service request; the result is obtained by matching "travel" with domain ontology and retrieving the related concepts. Any service request raised is parsed syntactically and semantically to identify the keywords .The service selection algorithm uses the keywords and conditions the presence of functional and nonfunctional properties of the web services[22]. We take functional properties of request and web service then we calculate the similarity between them. Name and textual description of request and services are matched using syntactic similarity function whereas inputs and outputs are matched based on semantic similarity function. In exiting three algorithm are interrelated keeps on decomposing the abstract goal into a set of more concrete tasks as the algorithm. traverses deeper in the path. In the second iteration of our algorithm, our algorithm identifies tasks, such as "Transportation", "Accommodation", "Tourist Attraction", and "Car Rental". In the third iteration, our algorithm further refines Accommodation task with more specific tasks, such as "Budget Hotel" and "Luxury Hotel". However, the entities, such as class "Bus" and "Air", do not have sufficient information (e.g., *attributes, subclass, sub-component, equivalence* entities, or *instances*) to discover new services[17]. Therefore, their parent node "Transportation" is not further decomposed to more specialized services

When these three algorithms are integrated to achieve the goal, some inconveniences are identified here and summarized as follows.

1. The semantic matching cannot be achieved as a single service.
2. Selected services with Qos are not based on the user constraint.
3. To generate composition plan are not supported automatically.
4. Qos service is not suitable for runtime.
5. It does not provide the optimal value in the ranking based algorithms.

Existing three algorithms are interrelated to travel concept for user goal . All three tasks are identified in the travel program 1.Transport 2. Accommodation 3 tourist attraction.

3.1 Identifying Task algorithm.

These tasks has constrains for user requested annotated in ontology. Using identify task algorithm to identify the user defined constraints

Algorithm A **Identifying Task List**

Input: Ontology model for the goal
Output: A set of task associated with services
Initiate: var E= the entity which match the gold description
Procedure identifyTask (var E)

```
-----------------------------------------------------------------------
{
```

```
1.    if E does not have attributes, direct subclass, sub-
components (described by part of relation), equivalent
entities, or instances
2.    {
3.    Return;
4.    Use equation (1) and (2) in search service for E in
      (ws_repository)*;
5.    If (the number of matching services  >  0)
6.    {
7.    Associate the matching service to E, and convert E
      as a task //  the ad-hoc process
8.    Output task E;
9.    }
10.   Set(Er) = e0  which have a direct relation with E
11.   If (the size of Set(Er) == 0)
12.   {
13.   return;
14.   }
15.   for each element Ei in Set(Er)
16    {
17.   identifyTask(Ei);
18.   }
19.   }
```

Algorithm B. **Searching for web service**
Searching to the web service is available for user request and to response available in UDDI registry.

Input: Ontology entity e0
Output: relevant service list
Procedure SearchServices ( )

```
1.    {
2.    entity (e0) = {e0} U {$e_1,e_2,e_3,\ldots e_m$} U attr ($e_0$)
3.    attr ($e_0$) = {$a_{e01,}\, a_{e02},\ldots a_{e0p}$}                    (1)

4.    entity $e_i$ = {$s_{i1},s_{i2},\ldots.s_{in}$}

5.    entity_keywords ($e_0$) = entity ($e_0$) U ($U^m_{i=0}$ syn ($e_i$))
//  each entity has own set of synonyms

6.    ws-keywords($s_i$) = {$t_1,t_2,\ldots.t_z$}
      //  where sj is the service and ti tag of service

7.    entity-keywords($e_0$) = ws-service ($s_j$)
8.    SIM= (#match keywords) l n
       //  n  is the number of tag description
9.    Using the formula (2) and (3) using in sim
11.   sort Sim (Rws1, Rws2, Rws3……Rwsn)
      //   sort the relevant service based on similarity
      degree
12.   if (Rws1 = Rws2)
      // Rws1,Rws2 are relevant services in service
      repository
13.   {
14.   Sort Qos (Rws1, Rws2)
15.   return sort result  (Sort list)
16.   }
```

Algorithm C. Semantic matching (user define in ontology match with web services)
The semantic matching algorithm is used to verify between user request and web services with similarity functions.
Input: Goal Description, Task Description (option)
Output: Matching Ontology

Procedure Search Onto ( )

```
1.    {
2.    var OntoSet = null;
3.    gd = gd U  { syn ki in gd } keywords in goal
      description with synonyms

4.    td = td  U{  syn  ki  in td  } keywords in task
      description with synonyms
      K(G) = gd ;
      K(T)  = td;

5.    for  each ki in K(G)//  keywords in goal description
      //  Search for Ontologies with keywords
6.    If Sementic match ( );
7.    add match  in to OntoSet
8.    }
9.    end for
10.   If OntoSet == 0
11.   {
12.   return null;
13.   }
14.   If OntoSet == 1
15.   {
16.   return 1 in OntoSet;
17.   }
18.   else
      //  OntoSet contain more on Ontology
19.   {
20.     SIM( dec,Sort  OntoSet ( n )) in K(G) and K(T)
      //  sort the selected ontology from similarity
      frequency of  K(G)= gd, K(T)= td.
21.   return  Ontoset(top)
22.   end else
23.   }
```

4. Proposed Work and its Algorithm

A query for the goal for planning a trip can be expressed using keywords, such as “Trip” or “Travel”. To derive the tasks that achieve the specified goal, we analyze the semantic meaning of the specified goal using ontologies. Ontologies capture the information related to particular goals using expert knowledge. For example the ontology for the concept “Travel” lists relevant concepts, such as “Flight”, “Hotel

Reservation", and "Tourist Attraction". To have a better understanding of the specified goal, we search for existing ontologies that can expand the meaning of a specified goal. Furthermore, we provide an algorithm that analyzes the identified ontology to dynamically discover services and compose to achieve the specified goal.

During the semantic matching a single web service cannot support to achieve a task and it cannot to achieve user goal also. The exiting three algorithms are integrated have to implement with Web service composition to achieve for this goal. Web composition is based on quality of service. The Qos based on more than constrains it may me automatic composition or dynamic composition These algorithms are implementing or enhancing algorithm in rank based algorithm are used to find the optimality of user goal. The implemented algorithm can be used to support automatic or dynamic and also get optimality rank based for user goal. These solutions are called ***Pareto solutions***. The QoS-aware composition based on run-time values. They supported to dynamic composition problem prior to execute the services, it is necessary to find optimal composition.

4.1 Web Service Selection (the problem identified area)

The current web service architecture and semantic web efforts address here problem of web service discovery and web services selection. Discovery deals with finding a set of services that corresponds to a predetermined user request while selection deals with choosing a service between those that are discovered. Moreover, selection seems to be the main problem. In fact, if the discovery process is exhaustive, a very large number of services may be found. Due to the number of services, and consequently in many cases, a single service is not sufficient to respond to the user's request and often services should be combined through services composition to achieve a specific goal. The composite services is starting to be used as a collection of services combined to achieve a user's request. In other words, from a user perspective, this composition will continue to be considered as a simple service, even though it is composed of several web services. the problem of composing web services can be reduced into four fundamental phases: The first one is planning, which determines the execution order of the tasks, we consider here a task as being a service functionality or a service activity. The second one is discovery that aims at finding candidate services for each task in the plan. The third phase aims at optimizing services composition and finally the fourth concerns execution.

The discovery process returns a set of candidate services from which the subset of those belonging to the composition should be extracted according to non-functional criteria (i.e. cost, availability, reputation). In fact, discovery is a prerequisite for selection, but selection is the main problem The non-functional criteria are here characterized by the QoS model presented in each web service. The QoS model has more than one criterion to be evaluated. Thus, services composition can be considered as a multi objective optimization problem.

4.2 Problem Definition

Given a user request UR(T,QWV,C) , we need to find the best composite services among the list of services that satisfies the user request where

(i) T denotes a set of independent tasks , where ranges from 1 to and denotes the number of tasks,

(ii) QWV is QoS weight vector which contains user preferences over QoS criteria:

QWV= (qw1,qw2,qw3….qw6)

(iii) c denotes the constraints specified by the user.

For example, a travel reservation scenario which is a typical web service composition problem offers travel, accommodation, and local transport rental services to the customers. The user request consists of a set of tasks like booking flight ticket, reserving hotel rooms, and renting a cab. Atomic services like flight service, hotel service, and taxi service are assigned to each task in the user request. Users can specify the type of services and local constraints like QoS preferences, global constraints, and other constraints like the total amount the user wishes to spend for the trip.

Concerning our Travel problem For an example If a user wants to travel, it is not sufficient to book a flight, but she should also take care of reserving a hotel, renting a car, getting entertained, and so on. Such composition is carried out manually today, it means that the user needs to execute all these services one by one and these tasks can be time and effort consume. so consider that we can now have more than task to be executed and over a hundred candidate services; Thus, combining each task, respecting their restrictions and respectively finding the service to execute the tasks can be considered as a combinatory problem. Since we treat our services composition as a combinatory problem it requires optimization, so our Travel problem can be treated as an optimization problem. Optimization problems require basically two elements: a search space composed of potential solutions and an objective function to be optimized. The search space may be restricted by a set of constraints.

In our example of Travel problem. In order to achieve optimal compositions we defined four main objectives that should be optimized: cost, time, reputation and availability. Cost represents the price of a service execution and Time is the execution time of a service. Moreover, Availability is the probability a service is "alive" and Reputation is the trustworthiness of the service in a determined field. Another important feature is that in a multi objective problem we do not have only one optimal solution but a set of solutions.

Algorithm I : User preference based web service

ranking algorithm

```
/ UserReq: User request
// UserReqSTi: User request service type
// ti: Task involved in user request
// WSLi: List of web services in the registry
// Si: Web service
// SLi: Search List
// FLi: Filtered List
// QRLi: QoS Ranked List
// QWV: QoS weight vector
// QVW= ($qw_1$,$qw_2$,$qw_3$,$qw_4$,$qw_5$,$qw_6$)
// $qw_1$: Cost Weight
// $qw_2$: Success Rate Weight
//$qw_3$: Frequency Weight
// $qw_4$, : Response Time Weight
// $qw_5$,: Reputation Weight
// $qw_6$ : Availability Weight
//RSLi : Ranked Services List
Begin
   (1) For each task ti in UserReq
   (2)   Discover(WSLi , UserReqSTi)
// check semantic matching

(3)      For each service Si in SLi do

   (4)         If( Si.Availability = = true)
   (5)         SL.add(Si)
   (6)         End if
               Else
 //     Call Web service composition
   (7)      Web service composition()
   (8)       End For
   (9)   QoS based Service Selection( SLi)
   (10)   Compute QoS Rank( FLi)
   (11)   Final Rank based Sorting (QRLi, QWV)
   (12) End For
   (13) Return RSLi
End
```

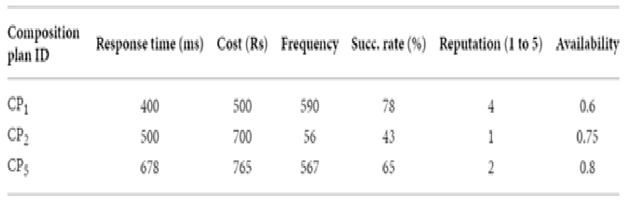
Algorithm II: QoS aware web service composition algorithm (UR (T,QWV,C), RSLi)

```
Begin
   (1) Rank Services
      (1.1) Perform UPWSR for each task ti in T
      (1.2) Save the RSLi for each task ti in T
   (2) Store each RSLi in task tables
   (3) Compute Service Composition (SC) table
      (3.1) Generate all possible Composition plans by
taking Cartesian product of all the
task tables obtained in Step (2)
      (3.2) Save the Composition plans (CP) in Service
Composition Table
   (4) Calculate QoS Aggregated value for each CP in
Service Composition and save in
Composition Plan List (CPL)
   (5) Constraint Analyzer
      (5.1) Perform Constraint Analyzer(SC, C) for each CP
in CPL
      (5.2) Save composite services that satisfy constraints in
Filtered Composition Plan
         List (FCP)
   (6) Pareto Optimal based Selection
      (6.1) Perform Pareto Selection(FCP)
      (6.2) Save Composition Plans after filtering in Pareto
Optimal based Selected
      List (POSL)
   (7) Compute Aggregated QoS Rank for each CP in POSL
      (7.1) Evaluate all the Rank for each CP in POSL
      (7.2) Save the CP with Rank in POSL
   (8) Calculate Final rank(POSL, QWV)
      (8.1) Compute Final rank for all CP in POSL
      (8.2) Sort and save the Composition Plan in Ranked
Composition Plan List (RCPL)
based on Final Rank
   (9) Execute all the Composition Plan in RCPL
   (10) Get feedback and update Rep(Si)
```

Semantic matching and web composition are Implementing in Algorithm I and II can be used to find an automatic and dynamic solution for ranking optimal goal based on user 's constrained.

4.3 Pareto Approach

Having several objective functions, the notion of "optimum" changes, because in MOP, the aim is to find good compromises ("tradeoffs") rather than a single solution. We can say that *xr* is Pareto optimal if there exists no feasible vector *yr* which decreases some criterion without causing a simultaneous increase in at least one other criterion.



Composition plan ID	Response time (ms)	Cost (Rs)	Frequency	Succ. rate (%)	Reputation (1 to 5)	Availability
CP_1	400	500	590	78	4	0.6
CP_2	500	700	56	43	1	0.75
CP_5	678	765	567	65	2	0.8

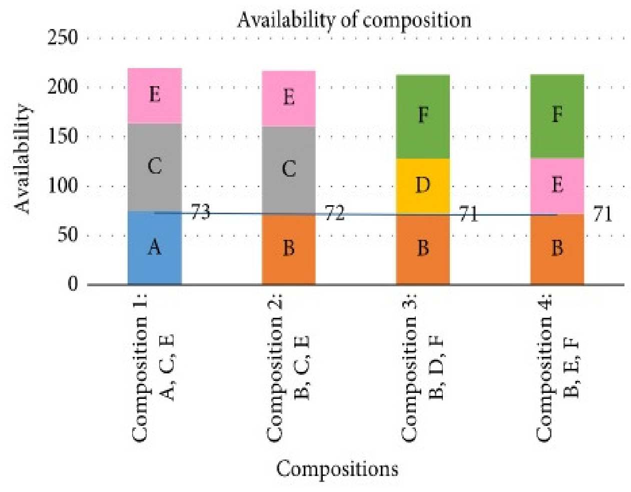
Table.1 Example of pareto optimal based selection

We have archived through this proposed enhanced ranking based algorithm to produce a web composition for web services. Composition plans from the composition plan repository are deleted periodically. Service providers can register their service manually in the service registry. This framework calculates Qos of those services dynamically.

Possible compositions from the available web services.

Compositions	Web services	Availability
1	A-C-E	73
2	B-C-E	72
3	B-D-F	71
4	B-E-F	71
5	A-C-F	83

Table.2 Generate compositions from the available web services



Composition generated for the service request.

5. CONCLUSION

In this paper, we provide an approach that hides the complexity of SOA standards and tools from end-users and automatically composes services to help an end-user fulfil their daily activities. We propose a tag-based service description to allow end-users to understand the description of a service and add their own descriptive tags. This paper presents a framework for composing Web resources in a personalized Web space. In the framework, Web resources are described by a unified description schema and are soap based web services. Heterogeneous Web resources hinder search engines and users to discover suitable Web resources for fulfilling users' goal of daily activities.The proposed approach dynamically composes web services and the composition plans are generate automatically.

In future we will improve our framework to allow a user to share access through mobile environment In our current implementation, the resource graph is created from the user's request and updated their profile for current web services, we plan to provide automatic approach to identify the relations among the Web resources and generate the resource graph for a given set of Web resources. We also want to design case studies to evaluate the performance of our framework for generating ad-hoc processes from a user's goal.

REFERENCES

[1]. R. Alarcón and E. Wilde. "RESTler: Crawling RESTful services." In Proceedings of the 19th international conference on World Wide Web, WWW '10, pages 1051{1052, New York, NY, USA, 2010. ACM.

[2]. A. Almonaies, J.R. Cordy, T.R. Dean, "Legacy System Evolution towards Service-Oriented Architecture", Proc. International Workshop on SOA Migration and Evolution (SOAME 2010), Madrid, Spain, pp. 53-62.

[3]. A. Alowisheq, D. E. Millard, and T. Tiropanis, "EXPRESS: EXPressing REstful Semantic Services Using Domain Ontologies." International Semantic Web Conference 2009: 941-948

[4]. M. Athanasopoulos and K. Kontogiannis, "Identification of REST-like Resources from Legacy Service Descriptions, WCRE 2010.

[5]. D. Beckett, B. McBride (editors), "RDF/XML Syntax Specification (Revised)," W3C Recommendation 10 February, 2004

[6]. B. Upadhyaya, F. Khomh. Y. Zou, A. Lau and J. Ng, A Concept Analysis Approach for Guiding Users in Service Discovery, IEEE International Conference on Service-Oriented Computing and Applications (SOCA'12) , Taipei , Taiwan.

[7]. Rajneesh Shrivastava, Shivlal Mewada and Pradeep Sharma, "An Approach to Give First Rank for Website and Webpage Through SEO", International Journal of Computer Sciences and Engineering, Volume-02, Issue-06, pp (13-17), Jun - 2014

[8]. M. P. Carlson, A H. H. Ngu, R. M. Podorozhny, L. Zeng, "Automatic Mash Up of Composite Applications," International Conference on Service Oriented Computing (ICSOC) 2008, Sydney, Australia, December 1-5, 2008, pages: 317-330

[9]. C. Engelke and C. Fitzgerald, “Replacing Legacy Web Services with RESTful Services,” WS-REST 2010 First International Workshop on RESTful Design

[10]. R. Ennals and D. Gay. “Building Mashups by Example”, Proceedings of IUI (2008)

[11]. R. J. Ennals, M. N. Garofalakis, “MashMaker: mashups for the masses,” Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM.

[12]. Yahoo Pipes: Rewire the web, http://pipes.yahoo.com/pipes/

[13]. Animesh Shrivastava and Singh Rajawat, "An Implementation of Hybrid Genetic Algorithm for Clustering based Data for Web Recommendation System", International Journal of Computer Sciences and Engineering, Volume-02, Issue-04, Page No (6-11), Apr -2014

[14]. I. Foster, C. Kesselman, J. Nick, S. Tuecke, “The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration,” Technical report, Global Grid Forum (2002)

[15]. Dublin Core Metadata Initiative, http://dublincore.org/, last accessed on September 9, 2010

[16]. Facebook, http://www.facebook.com/, last time accessed on August 22, 2011

[17]. R. Fielding. “Architectural Styles and The Design of Network-based Software Architectures”. PhD thesis, University of California, Irvine (2000)

[18]. Flickr, http://www.flickr.com/, last time accessed on Aug. 29, 2011

[19]. Gleaning Resource Descriptions from Dialects of Languages (GRDDL), http://www.w3.org/2004/01/rdxh/spec, lat accessed on October 12, 2010

[20]. B. Upadhyaya, R. Tang and Y. Zou. An approach for mining service composition patterns from execution logs. Journal of Software Evolution and Process; DOI: 10.1002/smr.1565, 2011.

[21]. Hassina Nacer, Djamil Aissani, “Review: Semantic web services: Standards, applications, challenges and solutions”, Journal of Network and Computer Applications, Volume 44, September, 2014 , Pages 134-151

[22]. Mallayya, D., Ramachandran, B., Viswanathan, S(2015) “An Automatic Web Service Composition Framework Using QoS-Based Web Service Ranking Algorithm”,The Scientific World Journal, pp.1-14.