# A Survey on Graph based Web Service Discovery and Composition Techniques

**Monowar Hussain, Arnab Paul**

***Abstract— Web Service composition technique provides the features to users that an individual web service cannot perform. There are several web services available over the web for performing different tasks. When there is no unique service capable of performing user request, there must be some way to sufficiently compose basic services to satisfy the user's request. Now it becomes very important to determine which service composition system is the most efficient one. This paper presents the requirement for service composition, the required technologies to perform service composition. It also provides several different graph based web service composition techniques. At service composition time, the composition of these services depends on the requester's inputs, outputs parameters and other non-functional parameters. Web service composition is a difficult task due to the asymmetric nature of results of the various services. In order to evaluate the best approach, various composition approaches were justified. We consider number of comparative parameters for evaluating the best composition plan.***

***Index Terms—Web Services, Semantic Web Services, Web Service Discovery, Graph Based Web Service Composition.***

## I. INTRODUCTION

Web services are well defined, self described and reusable software components that can be used over the internet with the help of technologies like SOAP (Simple Object Access Protocol) as a communication protocol, WSDL (Web Service Definition Language and Universal Description),UDDI (Discovery and Integration) that provides a mechanism to find existing web services. A web service can be defined as atomic or non-atomic (set of related components) which can be accessed through interface over the internet. Web services are loosely coupled fashion, allows ad-hoc, dynamic binding and are reusable software components. Web services can be classified into three categories and three entities based on their properties. The categories are registration, discovery and binding; and the entities can be defined as service requester, service provider and the repository (UDDI) [1]. The roll of web service repository where the service provider publishes is one of the most important aspects of the web that can reduce the overhead of service provider and explore the business. In present scenario web services are described in the form of WSDL format and it contains the syntactic description only.

**Manuscript Received on June 2014.**

**Monowar Hussain,** Department of Information Technology, Triguna Sen School of Technology, Assam University, Silchar, Assam, India.

**Arnab Paul,** Department of Information Technology, Triguna Sen School of Technology, Assam University, Silchar, Assam, India.

It only reflects the structure of the data which communicate between service requester and service provider over the web, but is unable to describe the semantic meaning of the data. And this leads the automatic web service composition difficult. In automatic web service discovery and composition, the semantic description and execution order of web services are important.

## II. DEFINITION OF WEB SERVICES AND SEMANTIC WEB SERVICES

### A. Web Services

Web services are independent, modular units of application logic which provide business logic to other applications via Internet. They allow communicating with business partners and their processes by means of a stateless model of "atomic" synchronous or asynchronous message passing. Web service interactions occur with the help of two specification languages: the Simple Object Access Protocol (SOAP) and the Web Service Definition Language (WSDL). They are platform and language-independent communication protocol that define an XML-based format for web services to exchange information over HTTP by using remote procedure calls. WSDL is an XML-based language which defines the interface that a web service exhibits in order to be invoked by other services. WSDL thus provides a functional description of web services consisting of inputs, outputs and exception handling [15].

### B. Semantic Web Services

The semantic web provides functional and non-functional description of web services which models the pre-conditions and post-conditions of the web service so that the determination of the domain can be logically inferred. It relies on ontology's to formalize the domain constraints what are shared among services. The goal of semantic web especially in present scenario regarding semantic web services is to fully automate the web services lifecycle. The semantic web considers the World Wide Web as a common connected data repository where web pages are remarked with semantic annotations. These annotations describe about web resources and their properties described in the RDF (Resource Description Format) [2] and the OWL-S ontology to describe further interaction and/or properties like equivalences, lists, and data types. With the semantic web technology, it is possible to write realistic and powerful applications that use annotations and appropriate inference engines to automatically discover execute and integrate web services [3].



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication Pvt. Ltd.*

## III. CURRENT WEB SERVICE COMPOSITION TECHNIQUES

### *A. Manual or Static Composition*

The static means that the service requester should build an abstract process model before beginning the composition; the abstract model includes a set of tasks and their data dependency. Each task contains a query clause which used to search the particular web service which will satisfy the task. For static composition there are two possible approaches available, *viz*, web service orchestration and choreography. In *orchestration*, existing web services are composed with the help of a central coordinator, which is responsible for invoking and composing the single sub-activities. In *choreography*, there is no assume of a central coordinator, but it defines complex tasks via the definitions of the conversation that should be undertaken by each participant. In static composition, the aggregation of service is done at design time and composition is performed manually, means that each and every web service is executed one by one in order to achieve the required goal. This type of composition is not flexible [4].

### *B. Dynamic Service Composition*

Web services are designed to support interoperability between different applications. Web services are platform independent as well as language independent. Because of these properties, the interfaces of web services allow an easy integration of heterogeneous systems. UDDI, WSDL and SOAP define standards for service discovery, description, and communication protocols. These web service standards however do not deal with dynamic composition of existing services. Business Process Execution Language for Web Services (BPEL4WS) focuses on representing composition where flow of the information and the binding between services are known a priori [17]. Dynamic composition of web services are more challenging problem. In particular, when a user request cannot be satisfied by a unique service; but the existing services can be combined to fulfill the demand. The dynamic composition of services requires the locality of services based on their capabilities and the recognition of those services that matched and could participate in composition, as described. The full automation of web service composition process is still an area of ongoing research, but accomplishing this aim with a human controller as the decision maker has already been achieved. The main problem for full automation of service composition is the asymmetry between the concepts that people use and the data that computers interpret. This can be overcome by using semantic web technologies [4]. Many approaches of web services composition methods have been proposed in recent years. In this portion, we discuss a brief overview of some techniques that deals with the web service composition. We survey only those techniques that use service dependency concepts and graph models. We can define dependency as whenever a web service receives some inputs and provides some outputs; the outputs are somehow related or dependent on the received inputs. By using a graph model, the properties of existing web services can be defined in terms of their input-output parameters, as well as semantic description about the web data. A graph contains set of vertices or nodes and set of edges that link pairs of vertices. A graph may be directed or undirected from one vertex to another. In a weighted graph, each edge is associated with a weight (some number). The dependency graph is used in web service composition to find out those web services which are participating in web service composition to satisfy a user's demand. Most of the composition methods that are based on graph theoretic approaches build web service dependency graphs dynamically. Graph search algorithms are used for traversing the dependency graphs in order to compose services. These methods can be distinguished based on how they search the dependency graph. A*, BF*, Dijkstra, Floyd Warshall, Forward chaining, Backward chaining, Bidirectional, BFS, DFS, Ford Fulkarson (Maxflow), Greedy search algorithms are examples of the most familiar search algorithms.

## IV. COMPOSITION ISSUES

In 2005, *Altheya Lang and Y.W.Su* [5] presented a model for web service composition based on AND/OR graph, and a graph search algorithm for searching the graph to find out the composite service(s) that satisfies a user request. For a given service request that only can be fulfilled by a composition of web services, their algorithm find the service categories that are relevant to the request and dynamically create an AND/OR graph to grasp the functional dependencies among the web services of these service categories. The graph is changed based on the information reflected in a request. The search algorithm is used to search the changed AND/OR graph for a minimal and adequate composite service template that satisfies the service demand. The algorithm can be executed repeatedly on the graph to find out different templates until the result is considered by the service requester. In 2011, *Elmaghraoui* [6] proposed a model for web service composition based on graph. They presented a solution for minimizing the computation effort in web service composition. They represented the semantic relationship between the participating web services through a directed graph. Then, they computed all pair shortest paths using a new version of the Floyd-Warshall's algorithm. Semantic similarity is referred as the degree of matching between concepts. To compute the semantic matching among services, they use subsume reasoning as originally proposed by Paolucciet al [7]. Subsume logic checks whether a concept is more general than other. Given two web services defined by vertices Vi and Vj this reasoning allows evaluating the degrees of similarity between the services using the scales: equivalent, subclass, subsumes and the below rules:

*a)* *Exact match:* If the outputs of Vi and the inputs of Vj are fully matched.

*b)* *Plug-in match:* If the output of Vi is a sub-concept of the input of Vj (Vj subsumes Vi)

*c)* *Subsumes match:* If the input of Vj is a sub-concept of the output of Vi. (Vi subsumes Vj)

*d)* *Fail match:* Neither subsumption nor equivalence relation between Vi and Vj. Thus, we link an edge



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication Pvt. Ltd.*

connecting vertices Vi to Vj if the degree of similarity between the outputs of Vi and the inputs of Vj is fall under above three condition. Weight of the graph is calculated based on degree of semantic similarity and some QoS parameters.

The work of *Mahmoud, Bettahar* and *Saidi* published in 2013 [8] can be considered as significant one. They proposed a model for automatically composing web services with the help of the directed graphs. The graph also describes the Web services, and the ordering of web services execution. In contrast, the user query, defined by a set of inputs and outputs parameters, can be stated as a directed graph composed of web services. They used web service as a function: Web service (Parameters, State-of-the-world), where parameters are input, output and state of the world is pre-condition and effects. *Hashemian et al.* [9] The authors uses dependency graph to store the I/O dependencies between existing Web services, and then composition of services is made by using a graph search algorithm. In their graph, each service and I/O parameters are denoted as a vertex; service's input/ output are denoted as incoming and outgoing edges, correspondingly. Here the authors considered the dependencies between input and output parameters only but they are not considering semantics description, thus they cannot guarantee that the produced composite services satisfy the requested functionality correctly. *Talantikite et al*. [10] proposed a method where a network of services that are pre-computed, stored and connected by their I/O parameters. The relationship between services is constructed by using semantic similarity functions depending on ontology. They represented the relationship among services using a graph structure. Their model used the backward chaining along with depth-first search algorithms to find the sub-graphs that contain relevant services which satisfy the requester demand. They proposed a solution to select the minimally composite services. However, their work constructs the graph at composition time which leads to computational overhead. The proposed approach of *Arpinar et al.* [11] used the concept of graphs for web service composition along with semantic similarity. They considered the weight associated edges and applied Bellman-Ford's algorithm for computing the shortest path. Weight of an edge is calculated by combining execution time of a service and input/output similarity. But nowhere have they mentioned anything about the non-functional parameters of web services. *Aydogan, H. Zirtiloglu* [12] used the backward chaining method along with depth first search algorithm to find the required services for a complex request. Their solution for web service composition is slightly abstract and does not clearly mention the execution plan of the algorithm. *Gekas et al* [13] developed a service composition task as a multilink graph network having no size limits and they dynamically observe the network structure to derive useful heuristics to instruct the composition task. Web services are described through a graph network and graph is constructed dynamically during the composition. For minimizing the graph searching time, a set of heuristics are used. But the authors claim that graph construction during composition is very inefficient in term of computation and thus it limits the scope of applicability of graph-based models towards web service composition problem. Nacera *Temglit* and *Ahmed Nacer* [14] proposed a basic work on how to build a flow graph of services and how to discover all possible plans of service composition from the graph satisfying a functional need of users formulated as inputs outputs parameters. The most interesting aspects of their solution are: the user does not have to specify the composition schema or participant services in his query; the user has to provide only Input and Output parameters of the desired service using his own vocabulary. The composite service is discovered dynamically and transparently for the request. Their proposed approach of composition can answer dynamically the known and unknown service processes requested by a user. This is important in web context where user requirements vary and the availability of services is not guaranteed. They implemented an index to enhance the service searching time and hence the response time. In parallel, they investigated the way of making a graph database to store the service flow graph. Graph database is often faster than relational database for associative data sets (graph data model) and they can scale more easily to huge data sets because they do not need expensive *join* operations. They mostly eliminate the problems of memory management.

Their matchmaking function is based on two concepts:

a) In chaining two service interfaces by finding Outputs-Inputs matching; this is called horizontal matching (See Figure. 1).
b) To retrieve two functionally similar services: user required service $S_R$ and available service $S_P$, by finding Inputs($S_R$)-Inputs($S_P$) and Outputs($S_R$)-Outputs($S_P$) matching; this is called vertical matching (See Figure. 2).

In horizontal matching, $S_1$ match horizontally $S_2$ when some (for inclusion) or all outputs of $S_1$ are matched (exact or subsume) by all inputs of $S_2$ (Shown in Figure. 1). In vertical matching, $S_R$ match vertically $S_P$ when all the outputs of $S_R$ are matched by all or some (for inclusion) outputs of $S_P$, and some or all inputs of $S_R$ are matched by all inputs of $S_P$ (exact, plug-in). This criteria guarantees that the published matched service satisfies the need of the searched service, and that the searched service provides to the matched service all the inputs it needs to operate correctly. So according to this assumption, matching function distinguishes four degree of valid matching*:*

*Exact, Plug-in, Exact inclusion, Plug-in inclusion* (Shown in Figure. 2).

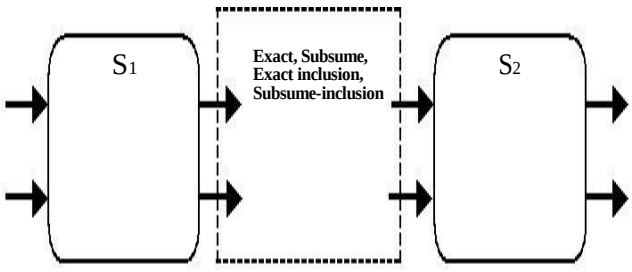


Figure 1. Horizontal Matching



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication Pvt. Ltd.*

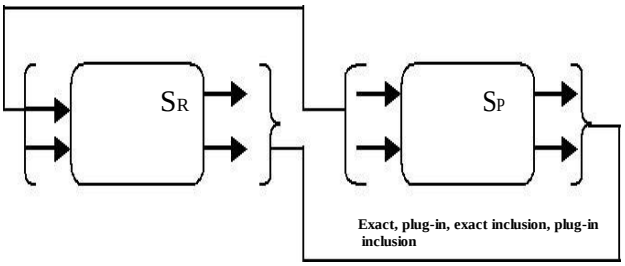


Figure 2. Vertical Matching

This guarantees that the first service provides to the second service all the inputs it needs to operate correctly in service composition context. Semantic similarity measures evaluate the semantic proximity between web services (to which the terms of queries and service parameters are attached). For this purpose they used Zargayouna technique for calculating semantic similarity indices of matching services. *Chan Oh* [15] proposed a graph based model in which they used bloom filter [1] which is a simple randomized data structure with minimal space for representing a set in order to support membership queries efficiently. In this regard, they used the concept of hash function whose time complexity is O(1). For graph searching, they used BF*

**Table 1. Evaluation Table**

| Authors | Algorithm used | QOS Awareness | Advantages | Disadvantage | Composition Pattern | Semantic Capability |
|---|---|---|---|---|---|---|
| Altheya Lang and Y. W. Su, 2005 | AND/OR (Explicit) graph search algorithm | No | 1. They used an admissible graph search algorithm, so it guarantees optimal solution, if one exists. 2. Requester has the flexibility to choose. | Required high analysis | Semi-automatic | No |
| Elmaghraoui | Floyd-Warshall algorithm | Yes | Requester gets web services having low cost and higher performance | Time complexity is high($O(n^3)$) and negative weight is not considered | Automatic | Yes |
| Mahmoud, Bettahar and Saidi, 2013 | Not defined clearly | No | Uses some preconditions for generating optimistic composition plan | × | Automatic | Yes |
| Hashemian | BFS algorithm | No | Ordering of web service execution | Does not guarantee correct service composition. | Automatic | Yes |
| Talantikite et al. | Chaining algorithm | Yes | × | Time consuming | Automatic | Yes |
| Arpinar et al. | Bellman-Ford's algorithm | No | × | Non-functional parameters are not considered | Semi-automatic | Yes |
| Aydogan, H. Zirtiloglu | Backward chaining and depth first search | No | Provides a better service composition plan | Does not consider QoS and cost parameters | Not defined | Yes |
| Gekas et al. | × | No | × | Time consuming and limited applicability | Automatic | Yes |
| Nacer Temglit and Ahmed Nacer | DFS algorithm | No | Composition plan can answer dynamically the known and unknown service processes requested by the user | Non-functional parameters are not considered | Automatic | Yes |
| Seog-Chan Oh et al. | BF* algorithm | No | Good efficiency because of using heuristic function | If heuristic becomes zero then it works like Dijkstra algorithm | Not defined | No |
| Kun YUE et al. | Greedy Algorithm | No | Time complexity is good | Does not consider non-functional attributes | Automatic | No |

Algorithm which is based on A* algorithm. At each state, A* algorithm considers some heuristics-based cost to pick the next state with the lowest cost. Combining this idea with Bloom Filter, they tried to improve the efficiency and



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication Pvt. Ltd.*

accurateness of their composition approach. Disadvantage: if set h(n) = 0, then BF* algorithm degenerates to Dijkstra's shortest path algorithm. *Yue, Weiyi Liu* [16] proposed a model for web service composition based on type matching applying the Greedy algorithm. A web service can participate in composition plan only if the input type parameters are satisfied, and an atomic service can participate in the result composition plan only if the output type parameters contribute to the type list that is required by users. The mutual association degree between two atomic services is called affinity and based on this affinity value *(0<= affinity(A,B) <=1)* the web service graph is constructed. Affinity value is the weight between two nodes (web services). They used Greedy algorithms to compose the web services.

## V. CONCLUSION

In this work, we studied a number of papers which describe web service discovery and composition techniques based on graph theoretic approaches. Here, we have studied graph construction techniques, input-output matching techniques, weight calculation techniques, shortest path selection techniques of various approaches. We have also compared various web service composition approaches that are based on QoS and some non-functional parameters. We have also evaluated some effective parameters (i.e. semantic capability) which are described in the above table along with advantages and disadvantages of various approaches.

## REFERENCES

[1] Schahram Dustdar and Wolfgang Schreiner, "A Survey on Web Services Composition", Int. J. Web and Grid Services, Vol. 1, No. 1, 2005.
[2] D. Brickley and R. V. Guha, "Resource Description Framework (RDF) Vocabulary Description Language (Version 1.0): RDF Schema, 2004", Available: http://www.w3.org/TR/rdf-schema/
[3] Maurice H. ter Beek, Antonio Bucchiarone, and Stefania Gnesi, "A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods", In Proceedings of the Second International
[4] Conference on Internet and Web Applications and Services (ICIW '07),Washington, DC, USA, 2007 © IEEE Computer Society, doi:10.1109/ICIW.2007.71
[5] Narges Hashmi Rostami , Eshmaeil Kherkha and Mehrad Jalali, "Web Services Composition Methods and Techniques: A Review", IJCSEIT: International Journal of Computer Science, Engineering and Information Technology, Vol. 3, No. 6, December 2013
[6] Qianhui Althea Lang, "AND/OR Graph and Search Algorithm for Discovering Composite Web Services", International Journal of Web Services Research, 2(4), pp. 46-64, October-December 2005.
[7] Hajar Elmaghraoui, Imane Zaoui, Dalila Chiadmi and Laila Benhlima, "Graph based E-Government Web Service Composition", IJCSI: International Journal of Computer Science Issues, Vol. 8, Issue 5, No. 1, September 2011, ISSN (Online): 1694-0814
[8] M. Paolucci et al., "Semantic Matching of Web Services Capabilities", In First International Semantic Web Conference, Sardinia, Italy, 2002, pp. 333-347.
[9] Chaker Ben Mahmoud, Fathia Bettahar, Hajer Abderrahim and Houda Saidi, "Towards a Graph Based Approach for Web Services Composition", IJCSI: International Journal of Computer Science Issues, Vol. 10, Issue 1, No. 3, January 2013, ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814
[10] Seyyed Vahid Hashemian and Farhad Mavaddat, "A Graph-Based Approach to Web Services Composition", In Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05)
[11] H. N. Talantikite et al., "Semantic Annotations for Web Services Discovery and Composition", Computer Standards Interfaces, 31(6), 1108-1117, Elsevier B.V, 2009.
[12] I. B. Arpinar et al., "Ontology-driven Web Services Composition Platform", Inf. Syst. E-Business Management, 2005, 3(2):175–199
[13] Aydogan, H. Zirtiloglu, "A Graph-based Web Service Composition Technique using Ontological Information", 2007, Vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1154–1155.
[14] J. Gekas, M. Fasli, "Automatic Web Service Composition based on Graph Network Analysis Metrics", In Proceedings of the International Conference on Ontology, Databases and Applications of Semantics (ODBASE), Agia Napa, Cyprus, 2005, pp. 1571-1587
[15] Nacera Temglit, Mohamed Ahmed Nacer, "Graph Based Approach for Dynamic Discovery of Composite Web Services" In 2012 IEEE Conference on Open Systems (ICOS), Kuala Lumpur, Malaysia, Print ISBN: 978-1-4673-1044-4, 2012 © IEEE Computer Society, doi: 10.1109/ICOS.2012.6417635
[16] Seog-Chan Oh et al., "BF*: Web Services Discovery and Composition as Graph Search Problem", In Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05), 2005 © IEEE Computer Society, Print ISBN: 0-7695-2274-2, doi: 10.1109/EEE.2005.41
[17] Kun Yue, Mingliang Yue, Weiyi Liu and Xiong Li, "A Graph-Based Approach for Type Matching in Web Service Composition", Journal of Computational Information Systems 6:7(2010) 2141-2149, ©2010 Binary Information Press, Available: http://www.jofcis.com
[18] Antonio Bucchiarone, "A Survey on Services Composition Languages and Models", International Workshop on Web Services Modeling and Testing (WS-MaTe 2006)



*Published By:*
*Blue Eyes Intelligence Engineering*
*& Sciences Publication Pvt. Ltd.*