I.J. Education and Management Engineering, **2017, 4, 41-50**
Published Online July 2017 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2017.04.05

Available online at http://www.mecs-press.net/ijeme

Dynamic Composition of Web Services by Service Invocation Dynamically

Sumathi Pawar [a]*, Dr. Niranjan N. Chiplunkar [b]

[a] *Canara Engineering College, Mangalore-574219, India*
[b] *NMAMIT, Nitte, Karkala – 574104, India*

Abstract

The automatic Web service composition is one of the greatest challenges. The problem of unavailability of public UDDI is motivation to develop an interactive system to search for Web services dynamically using search engines. This paper presents an interactive system for selecting composable Web services automatically and dynamically according to the user requirement through search results of Bingo search engine. The methodology used here is searching for requested functions according to the user requested functional word in the Bingo search engine, finding the search precision by the support of the requested function in the search results, displaying operation elements and allowing the user to select required operation. If the user is unable to enter the input value of the Web services then searching for the Web services that resolves the unknown input. The search process is continued in many levels till the user gets satisfied and the suitable Web services resulted in this process results in the dynamic composition. A composition rule is framed to show the operations of the Web services that are composed according to the user requests during the run time. This is an interactive system, where the user can select required operation from the list of operations. The required parameters to invoke the Web services are filled during runtime automatically resulted into a dynamic invocation of the Web services. Nowadays, QoS information such as availability and response time are not provided by the UDDI due to the absence of UDDI. Therefore, this system tests such information by invoking Web services and qualitative Web services are used to generate composition plan.

Index Terms: Web services, WSDL, dynamic composition, static composition.

1. Introduction

The dynamic composition is the composition of Web services during run-time. In this research

* Corresponding author.
E-mail address: pawarsumathi@gmail.com

searching for the user requested functionality in the search engine as Web services and finding the unknown input values of these Web services as the output of other Web services at different levels of search resulted into the dynamic composition of Web services.

This system searches for user requested functional word in the Bingo search engine as WSDL of the Web services. From the retrieved result, only WSDL links will be extracted to check the availability of the requested functionality in the Web services.

The system processes these WSDL files and compares each functional word with service names of WSDL for the exact or partial match. The system also displays all operations of these matched Web services. Then the user is required to interact with this system by selecting particular operation.

The system also interacted with the user to ask the value of input parameter. If the user does not know the input then a search process will be performed to get the operation of Web services which results into required output that matches to the unknown input value. The result of this operation will be given as input to required service. This process will be continued till the user requested functionality and input requirements are met. The Web services involved in this process are used to generate a rule for composition plan.

The organization of the paper is like this. In section 2 literature survey is given, section 3 discussed the dynamic composition of Web services, Section 4 is focused on the different scenario for the composition of the Web services and section 5 discusses the experiments, section 6 about the analysis of results and section 7 about the conclusion and future work.

2. Related Work

In the existing paper [1] authors have given an approach for automatic Web service composition. In their paper, they have considered only input/output parameters of WSDL to match with the user request, whereas in this research, operation, input elements are checked against user request and a different methodology is used for dynamic service composition as given in section 3.4,5.

As given in the survey authors worked out with different approaches to Web service composition including [7] BPEL4WS for manual composition, [3] by mapping between 10 parameters and rule to select best composition path.

[4] Authors also used classifiers with text mining techniques to find the syntactic difference between service descriptions. But the “Bingo” search technique used in this research retrieves the semantically similar Web services.

[5] The linear temporal logic formula is used by authors with client specifications to discover the services. But the proposed research discovers the suitable Web services by WSDL processing.

[2]Authors also used AI planning techniques with OWL-S to find semantically matching Web services. But proposed system does not use OWL-S like static information.

[6]Genetic algorithm used by authors run endlessly till fixing the number of iterations for service composition. Causal Link Matrix (CLM) [8] given by the authors, provides a solid background for semantic Web service composition through AI techniques. But the proposed research retrieves semantically related Web services through Bingo search engine.

[17] An algorithm to efficiently select optimal union composition is Spatio-temporal union composition which is planned by considering new QoS criteria. During composition itself, this algorithm finds the lowest cost neighbor Web services. Authors proposed a variation of Dijkstra shortest path finding algorithm that is called as UnionComposition which minimizes the search cost. A directed graph is modeled by this algorithm, in which each vertex is Web service, which has associated space-time-attribute and edges have QoS attributes. They indexed the services using the spatiotemporal features in a 3D R-tree. The composable services are searched using location and time, which also justifies that coverage of composition should not change in space and time. But the Web services used here is hot spot services with assumptions that they are static in nature, therefore composition plan

generated here is static composition plan. But in the proposed research we are generating dynamic composition plan.

[18] A Declarative Framework for Self-Healing Web Services Composition (DISC) framework provides an approach for self-healing Web services which is the constraint-based declarative approach. This system has three stages of the composition such as Composition design, Instantiation and execution, and Composition monitoring. The monitoring of composition is helpful to avoid any deadlocks, and it gives a set of solutions for the composition process. Provisions of recovery actions such as re-planning to find alternatives are provided if it detects the violations.

Chen Wu and Elizabeth Chang [19] have indicated that "Public UDDI Business Registry - the primary service discovery mechanism over the Internet has been shut down permanently since January 12, 2006 due to several reasons".

It is observed from the literature survey that the absence of UDDI and lack of work in dynamic composition motivated the proposed system for dynamic composition of Web services using search engine.

3. Dynamic Composition of Web Services

The proposed research dynamically composes the Web services by using several steps from section 3 to section 6, are given in the pseudo-code below.

Step 1: Input functional word
Step 2: Prepare a query using this functional word and feed it to Bingo search engine by using JSoup.Connect API
Step 3: Extract only WSDL links from the retrieved results of Bingo Search Engine.
Step 4: Store these links in the retrieved_links of the results.
Step 5: For each WSDL link of retrieved_links Repeat
Step 6: Retrieve the WSDL description through the URL of the link from the online and store it in the WSDL file.
Step 7: Extract <service> element from the WSDL and compare it with given functional word.
Step 8: If it matches with given functional word partially or fully or semantically then store such WSDL link in MatchedURLlist.
Step 9: Display all operation elements of this matched service
Step 10: Allow the user to interact with the system to select an appropriate operation.
Step 11: Display the input parameters of the selected operation and allow the user to enter the value of input element.
Step 12: If the user does not know the input then pass this unknown input parameter name to search system as a functional word to search for Web services which are called as dynamic composition**.**
Step 13: Continue the process of dynamic composition until then unknown input is resolved by the search results and store the Web services involved in this process in the dynamic composition plan.
Step 14: Invoke all the Web services of the dynamic composition plan in Last-in-First-out basis and feed the required output as the input to its preceding service.
Step 15: Return the result of the First-in Web service to the user.

Endif
Endif
Endfor

Step 16: Stop

To get information about semantically related Web services, we need to search for requested functionality using following query in Bingo search engine by giving[9]

"Functional_word?wsdl"

This retrieves web result that has WSDL links as well as additional data as a result of the search query. From these additional data, we need to extract only WSDL links. The connect API of the internet is used to connect a user from the system to the internet. The search engine URL of "Bingo" search engine with requested functional word is executed from the system as given below.

Document doc = Jsoup.connect ("http://www.bing.com/search? q="+Functional_word+"%3Fwsdl"). get();

From the retrieved results, only WSDL links are retrieved using the statement given below.

```
Elements links = doc.getElementsByTag("a");
        //for loop to retrieve links present in the HTML page
        for(Element link : links)
                {String linkHref = link.attr("href");
                        urllist.add(linkHref);
                }
```

The output of this code snippet is different links available as URL location of WSDL and count of these links gives retrieved number of links. This retrieved WSDL information is to be processed to compare the requested functional word with <service> name element of WSDL. If it partially matches then this is the required information and we need to keep count of these required WSDL links.

The pattern match function facilitates the matching of the functional word to <service> element using partial match or exact match or semantic match.

[9] The Search Precision **or** support of requested term in the search result in the <service> element of Web service is computed in equation(1)

$$Precision(Support) = \frac{Number\ of\ Required\ WSDL\ links}{Number\ of\ \ Retrieved\ links} \quad (1)$$

The operation elements of matched Web services are displayed and the user is prompted to select the particular operation. After selection of operation from the list, the user is prompted to enter input value. If the user does not know the input then searching for this input element as output of operation element of same Web service is followed. This unknown input is given as functional-word to the Bingo search engine once again to retrieve Web services which resolve this unknown input and called as dynamic composition. For this purpose, the iterative search and invocation of services till it gives required output is performed which is explained with an example in the next section.

4. Scenario for Dynamic Composition of Web Services

The user or consumer of the Web service can give functional-word as the request to the proposed system.

For e.g. if the input given to this frame- work is word "Weather", then our system prepares the query and feed it to the Bingo search engine through Jsoup.connect API as given here

Document doc = Jsoup.connect ("http://www.bing.com/search? q="+Weather+"%3Fwsdl").get();

The extracted WSDL links from the search result are processed for comparing <service> element with the user requested functional word. The matched WSDL links with their search precision are given in figure 1 below.

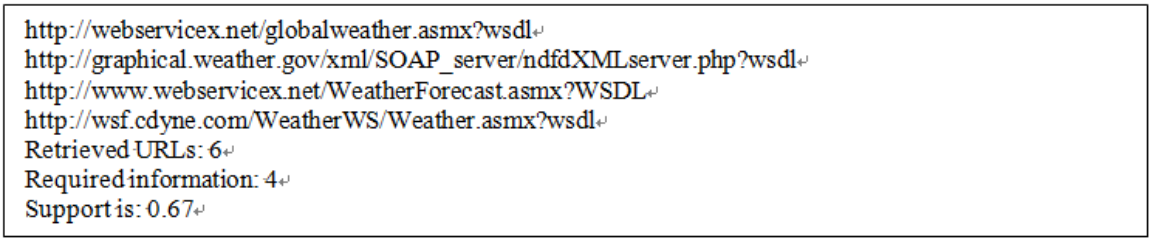

Fig.1. Matched WSDL Links to the User Request through the Bingo Search Engine

4.1 Scenario 1

The <service> element of this service "GlobalWeather" is compared with the requested functional word. If it exactly or partially matches then operation elements of that WSDL file is displayed to allow the user to select particular operation. For e.g. the operation elements of "Weather" service are

GetWeather
GetCitiesByContry

If the user selects "GetWeather" operation then WSDLHelper API class is used to extract <input> elements of "GetWeather" operation using the statement

Require_Input_List= WSDLHelperObject.getInMessageParts(Matched_operationName);

The <input> elements are "cityName" and "countryName". The user is allowed to enter "cityName". If the user is not knowing the "cityName" then this is searched as functional word in the online search engine i.e. Bingo search engine as

Document doc = Jsoup.connect ("http://www.bing.com/search? q="+ cityName +"%3Fwsdl").get();

If the user knows the input then after entering these values the Web service is invoked dynamically by filling the required parameter as given below in the different steps.

1. String wsdllocation= "http://www.webservicex.net/globalweather.asmx?WSDL";
2. String namespace = "http://www.webserviceX.NET";
3. String serviceName = "GlobalWeather";
4. ServiceFactory factory = ServiceFactory.*newInstance();*
5. Service service = (Service) factory. createService(new URL(wsdllocation),new QName(namespace,serviceName));
6. QName portName = new QName(namespace,"GlobalWeatherSoap");
7. QName operationName = new QName(namespace,"GetWeather");
8. Call call = service.createCall (portName, operationName);
9. Object[] params = {"Mangalore","India"};
10. Object weather= call.invoke(params);

The code snippet given above uses WSDL link, target name-space and service name to create service object as given in the steps 1,2,3,4,5.

The binding protocol as "GlobalWeatherSoap" and operation name as "GetWeather" to create the call object as given in the steps 6,7,8.

The invoke method of the call object is used to call the Web service with “Mangalore” and “India” as input parameters as given in the steps 9 and 10.

4.2 Scenario 2

The functional request “Weather” is also matched to “WeatherForecast” service partially as given in figure 1. Among the operations of this Web service, “GetWeatherByZipcode” operation is selected by the user. But the input element is “zipcode”, may not be known to the user. Therefore this unknown input is given as functional word, once again to the Bingo search engine as “zipcode?WSDL” to search for composable Web service. The extracted WSDL follows the same process as given in the section 3 and section 4.

The sequence of steps given in the psuedo-code of section 3 resulted in the search of “USZIP” Web service which contains four different operations. Among these list of operations, “GetInfoByCity” operation is selected by the user. The input of this operation is “usCity”. If the user is able to give the input, then required parameters are filled and this operation is invoked using call.invoke() method. The service invocation results into the list of 400 rows of zip-codes of different areas of XML data set.

The processing of the XML data set returned by the zip-code service invocation returns the zip-code of the required area.

This zip-code is given as input to the prior operation “GetWeatherByZipcode” of the “WeatherForecast” service. The service invocation of this service by filling required parameters, results into required output i.e. weather of particular zip-code.

The <operation> elements involved in resolving the unknown input are considered to build the rule as given in equation (2).

The syntax of the rule is given here.

If $OP_{nout} \rightarrow OP_{n\text{-}1in}$ U $OP_{n\text{-}1out} \rightarrow OP_{n\text{-}2in}$…… U $OP_{n\text{-}iout} \rightarrow OP_{n\text{-}i\text{-}1inn}$ (s) then OP1 U OP2 .... U OPn → Sreqout (S) (2)

Here i ranges from 1 to n-1 and S_{reqout} is required output parameter and OP1,OP2,OP3..OPn are composable operations of the Web services which feeds its output to the input of its prior Web service. Support (S) is support of composition plan which calculated in the equation 3.

$$S = \frac{\text{Number of Web services participated in the composition}}{\text{Total number of retrieved WSDL links}} \quad (3)$$

Thus the Web services which are resulted by equation(2) with non-zero value of support are considered for building the repository of composition plan to satisfy the user request. *S*ervices given in the rule (2) are needed to be invoked *sequentially*.

5. Results

Experiments are carried out with I3 processor with 2.20 GHz, 64 bit OS with 4GB RAM. The different methods of WSDLHelper class are used to process the WSDL. The results of following requests are shown in figure 2.

“weather?WSDL”
“Temperature?WSDL”
“Zipcode?WSDL”
“Curreny?WSDL”

As shown in figure 2, the proposed research has got the support of simple search as 66% due to the existence of requested functional-word in <service> element of similar web services. For "weather?WSDL" request the "Bingo" search engine has got six WSDL links from the Bingo search engine. After comparing with <service> elements, there are four WSDL links matching to the requested functional word. Therefore precision or support of the search is 4/6 that is 67%. Similarly for other requests the result is shown in the figure 2 given below.

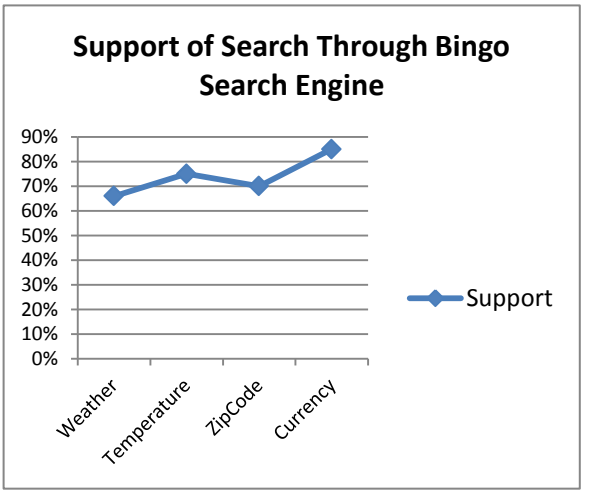

Fig.2. Support of Search through Bingo Search engine for different Search Requests

Figure3 shows the result of support of sequential composition plan. The search precision (support) of search results in searching the Web services towards unknown input terms and in forming the composition plan is already computed in equation 3.

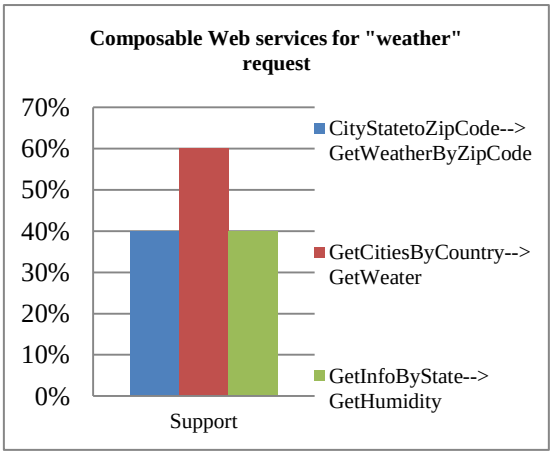

Fig.3. Support of Composition of Composable Search Request through Bingo Search Results

Figure 3 shows the result of support of sequential composition plan. The search precision (support)

of search results in searching the Web services towards composable Web services for "weather" request with different input are given in the equation 3. As shown in the result of composable Web services, before the operation "GetWeatherByZipcode" the operation "CityStateToZipcode" is need to be invoked. In another example before the operation "GetWeather", the operation "GetCitiesByCountry" is need to be invoked.

Therefore composition plan generated are

- CityStateToZipcode→ GetWeatherByZipcode
- GetCitiesByCountry→ GetWeather
- GetInfoByState→GetHumidity

It is explained in the scenario 2 of section 4.2 that, the existing online composable web services are very few. If the same implementation applied for other simulated composable Web services then, this research will be result into more number of results. In the online if more number of composable web services are published in the future, then the proposed implementation will dynamically compose these web services according to the user requests.

6. Analysis

[10] It is necessary to search for same requests dynamically using search engines because of dynamic nature of the web services. We have computed the search precision of search results and search precision may change during different searches because of dynamic nature of the Web services. The composition plan that gives highest confidence is considered for satisfying the user request.

According to the results of WOOGLE[11], it provides higher precision and lower recall. By considering parameter matching and concept matching WOOGLE obtains the recall as high as CONONLY only and precision as high as PARONLY only.

But Woogle aquires Web service descriptions through UDDI registries. Search is performed through the key-words, and offers the possibility to discriminate between inputs and outputs. This system does not depend on UDDI because of its absence. This system uses only Bingo search engine to acquire WSDL.

[12] VUSE is another example of specialized web service search engine which retrieves the WSDL. VUSE also depend on UDDI to retrieve WSDL and it retrieved eight WSDL links for "weather" as search query. But now a day all the WSDL links among those links does not contain the services. Therefore our system invokes the services and based on availability of the services it generates composition plan.

In some paper authors have used static templates and in some papers authors have used AI which requires the knowledge base of modular approach of sub-requests for different requests. But our methodology generates the composition plan without using any template and knowledge-base. It searches for availability of the requested functions in the WSDL of the available online Web services and filters the search results to reduce the workspace for further procedures.

7. Conclusion and Future Work

In this research, we composed the Web services dynamically by using search results of Bingo search engine because Google search engine does not allow to extract retrieved result of the search programmatically. In this system quality of service information is not considered because of unavailability of QoS information of Web services. But in this system QoS information such as availability and response time is tested after invoking the service. The Web services which are available and tested for quick response only used for building composition plan.

In this research, composition plan is generated in search process of unknown input and execution of Web services of the composition plan gives the final output. If there are no Web services available which resolve the unknown input, then this composition goes into infinite loop. To overcome this problem, user can provide finite number of iterations to repeat. As a future work the problem of infinite loop will be solved using required techniques, so that system will automatically come to a finite state.

Acknowledgements

I thank my guide Dr. Niranjan N Chiplunkar, for his great support and motivation. I also thank my previous guide Dr. Ashok Kumar A., who is passed away and his blessings is a great moral support for this research.

References

[1] Abdellah Kouider, Mohammed Erradi Hamid, Azzoune Algiers, Algeria, A Discovery Service for Automatic Composition of Web Services Oriented-Agent, IEEE INTERNET COMPUTING 2013, Pages: 33 - 35, DOI: 10.1109/WETICE.2013.16.

[2] Ourania Hatzi, Dimitris Vrakas, Mara Nikolaidou, Nick Bassiliades, An Integrated Approach to Automated Semantic Web Service Composition through Planning IEEE transactions on services computing -2012.

[3] Zhang MW, Zhang B, Liu Y et al. Web service composition based on QoS rules. Journal of computer science and technology 25(6): 1143–1156 Nov. 2010. DOI 10.1007/s11390-010-1091-6, Springer Science Business Media, LLC & Science Press, China.

[4] Walid Gaaloul, Karim Ba ̈na, Claude Godart, Log-based mining techniques applied to Web service composition Reengineering, SOCA (2008) 2:93–110, Springer-Verlag London Limited – 2008.

[5] Delnavaz Mobedpour Chen Ding ,User-centered design of a QoS-based web service selection system SOCA 7:117–127 Springer-Verlag London Limited -2013.

[6] M. C. Jaeger and G. MuThl, "QoS-based Selection of Services: The Implementation of a Genetic Algorithm," In Torsten Braun, Georg Carle, and Burkhard Stiller, editors, Kommunikation in Verteilten Systemen (KiVS 2007) Industriebetrage, Kurzbeitrage Workshops, March 2007, pp. 359-350, Bern, Switzerland, VDE Verlag, Berlin und Offenbach.

[7] S. Thatte,ed., BPEL4WS (Version 1.1), www.ibm.com /developerworks/library /specification/ws-bpel, 2003.

[8] F. Lecue and A. Leger, "A Formal Model for Semantic Web Service Composition," Proc. Leading the Web in Concurrent Eng.: Next Generation Concurrent Eng., pp. 385-398, 2006.

[9] Sumathi, Dr. Niranjan N. Chiplunkar, Dr. Ashok Kumar A. ,Dynamic Discovery of Web Services", Sumathi et.al " Dynamic Discovery of Web Services using WSDL", International Journal of Information Technology and Computer Science(IJITCS), MECS Publishers, IJITCS Vol. 6, No. 10, PP.56-62, DOI: 10.5815/ijitcs.2014.10.08, ISSN: 2074-9007 (Print), ISSN: 2074-9015 (Online), September 2014.

[10] Sumathi, Dr. Niranjan N Chiplunkar "Necessity of Dynamic Composition for Web Services", Paper presented at International Conference on Applied and Theoretical Computing and Communication Technology IEEE-2015.

[11] Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang. J., "Similarity search for web services", Proceedings of VLDB, 2004, pp 372-383.

[12] Ourania Hatzi, Georgios Batistatos, Mara Nikolaidou, Dimosthenis Anagnostopoulos, "A Specialized Search Engine for Web Service Discovery", International conference on Web Services Computing IEEE- 2012.

[13] Delnavaz Mobedpour Chen Ding ,User-centered design of a QoS-based web service selection system SOCA 7:117–127 Springer-Verlag London Limited -2013.
[14] Jiaxing Shang, Lianchen Liu, Web Service Composition based on Complex Networks pp 208-213 International Conference on Service Science – 2013.
[15] WANG Yi, SUN Hui-juan, The application agents and web services based on ontology, I.J. Education and Management Engineering 2012, MECS publishers, DOI: 10.5815/ijeme.2012.01.03
[16] Aparna Vijaya, Neelanarayanan V, A Model Driven Framework for Portable Cloud Services: Proof of Concept Implementation, MECS Publishers, PP.27-35,DOI: 10.5815/ijeme.2015.04.04,2015
[17] Azadeh Ghari Neiat(B), Athman Bouguettaya, Timos Sellis, Spatio-Temporal Composition of Crowdsourced Services, ICSOC 2015, LNCS 9435, Springer 2015, pp. 373–382.
[18] Ehtesham Zahoor, Kashif Munir, Olivier Perrin and Claude Godart, An Event - Based Approach For Declarative, Integrated And Self - Healing Web Services Composition, International Journal of Services Computing (ISSN 2330-4472) Vol. 1, No. 1, IEEE 2013, 13-24.
[19] Chen Wu and Elizabeth Chang, Curtin University of Technology, Australia: " Searching services on the web: A public Web Services discovery approach" 2008 IEEE Int. conf. on Internet Base System.

Authors' Profiles

Prof. Sumathi pursuing her PhD under the guidance of Dr. Niranjan N. Chiplunkar in Computer Science and Engineering Department, NMAMIT, Nitte, Karkala, VTU, University, India. She did her M.Tech from NITK, Suratkal, Karnataka, India and presently she is working as Associate Professor in Computer Science and Engineering Department of Canara Engineering College, Mangalore. India. She has 15 years of teaching experience. Her research area includes Web Services, Cloud Computing and Data Mining. She is associated with ISTE and also published papers in National and International conferences as well as in journals.

Dr. Niranjan Chiplunkar did his BE(E&C) from NIE Mysore in 1986, M.Tech(CSE) from MIT, Manipal in 1991 and Ph.D. in computer science and engineering from University of Mysore in 2002. His areas of interest include "CAD for VLSI", "Web Services", "Embedded Computing" and "Computer Networks". He is a member of IEEE, Computer Society of India and Indian Society for Technical Education. He is a fellow of Institute of Engineers(India). He has more than 33 years of teaching experience. He is currently the Principal and Professor in Computer Science and Engineering at NMAM Institute of Technology, Nitte, India. Prof. Chiplunkar has successfully completed four research projects grants and presented more than 70 technical papers in National and International Conferences and journals. He also written two text books and has been awarded with "Excellent Achievement Award" from Centre for International Cooperation on Computerization, Govt. of Japan in March 2002. During 2007, he has been given "Distinguished Alumnus" award by the Manipal University.

How to cite this paper: Sumathi Pawar, Niranjan N. Chiplunkar,"Dynamic Composition of Web Services by Service Invocation Dynamically", International Journal of Education and Management Engineering(IJEME), Vol.7, No.4, pp.41-50, 2017.DOI: 10.5815/ijeme.2017.04.05