Implementation Of Agent Based Web Service Composition Using Dependency Relation Algorithm

Francis Michael R.T, Siva Pragasam. P

Researh Scholar, Dept. of Computer Science, Bharathiar University, Coimbatore, TamilNadu, India
Associate Prof., Dept. of Computer Science, Sri Vasavi College, Erode, TamilNadu, India
Email: francismike009@yahoo.com

ABSTRACT: In the modern world of web service community expands its way with increasing the number of web services to facilitate their users. This expansion also comprises several problem like identification, selection and composition of web services. We have proposed and implemented a method that may give the solution to such problems. This method can be very useful when number of web services are very high in number. It is an agent based model that uses the dependency relation algorithm to identify the web services which are participate in the composition. The experimental part of the article shows that there is an improvement in the approach.

Keywords: Web service agent, Dependency relation, Web service composition, SOA agent, Web services parameters

1 INTRODUCTION

THe development of distributed application environment intended to realize the capability of agent based applications [1]. Most of the existing approaches have more pitfalls. The necessity of developing agent based system is to enable the autonomous composition of web services, dynamic and intelligent in nature. To make the agent as dynamic and intelligent it must incorporate the attributes such as integrity, consistency, undependability and adoptability. Integration of SOA and SOC strengthen the advantages of developing an agent based web service composition [2]. One of the emerging application architecture SOA provides the software services are elementary known as web services (WS) can be composed and used it to achieve the complex tasks. WS are the methods created, published and accessed over the web. The UDDI, WSDL and SOAP are involved in this invocation process [3-6]. Fig 1 shows the messages of SOA.

Fig 1. *SOA and it's messages*

SOA employs various approaches to compose the WSs they are semi-automatic and fully automatic. Choosing of this approaches depends upon the domain and the respective process model. Web services are heterogeneous and loosely coupled over the SOA. One of the important characteristics of this type of the system is that adoptivity of the ad-hoc changes in the components of distributed system. The primary task of an agent based WS composition model is searching the services in the UDDI and providing the composition of services for the service requestors. This article implements a multi-agent model and the contents are organized as follows, Section-I describes the introductory part, Section-II reviews the existing approaches, Section-III implements the relevant model and evaluates the results and Section-IV concludes the approach.

2 OVERVIEW

The need of an agent in the WS environment is to separate the service domain from the service requester and reduce the bottlenecks of the system [7]. WS agents are the one providing the required services by processing the communication messages between the service provider and the service requester. Nicholas Gibbins et al. proposed an agent based architecture called central broker manages the messages between the system components. The agent handles the multiple data sources called as heterogeneous data streams [8]. Jonathan D et al. implemented the agent based WS using April Agent Platform (AAP) and DAML + OIL. This model collects the stream of data from the web which includes the information of web pages. The persistent of the service instances fully depend upon the database [10]. This model is limited is limited to the underlying domain. The scope of the future WS agent system adopts the semantic constructs which includes the ontologies, DAML, RDF and XML [11]. WS agent acts an interface between the service repository and service invoker. The agent can establish the communication when service request is made. It also notifies the updates when there is a change in the service repository. It also manages the current web standards and protocols. Agents are also take care of the privacy enabled contents and the role based access if the WSs are involved in the secured transaction. The scalability of the environment is fully depend upon the agent it also implements the load balancing to ensure QOS. The QOS attributes are playing a vital role on selecting appropriate service [12]. QOS enabled multi agent systems are involved in the integration of the dynamically selected WS. This system collects the user QOS preferences based on that WS selection is made [13]. The service agents are responsible for several roles deals with service requests and service responses. For the given request the appropriate services and their corresponding agents are identified to prepare the composition plan. Shanliang P et al.

proposed a multi agent model dealt with the service selection planning problem. The service planning is done by considering the relationships among the different service consumers and providers [14]. Ontology enabled intelligent software agent is a multi-agent system. It processes the user requests based on the functionality and OS requirements. The similarity between the services are calculated by the input and output parameters [15]. Cosine similarity between the web services are determined to match the web services with respect to the user's request [26]. Automated score based indexing may not suitable for comparison based model. The semantic similarity measures have been introduced by to measure to measure the similarity between the words and sentences [27]. Features and limitations of the various agent models reviewed and are listed in the Table 1. The general characteristics of the agent listed in the Table 2

TABLE 1
Features of existing WS-agent models

e	Models	Features	Additional requirements
1	Centralized broker [8]	Domain specific, heterogeneous data stream, dynamic agent	High speed query processing
2	Fujitsu EO prototype [10]	Supports B2C environment, AAP and FIPA, applicable to large scale environments	Supports only underlying domain
3	Multi agent framework (WSAF) [13]	SA exists between service consumers (SC) and service providers (SP)	Aware of QOS requirements
4	SWSCPA [14]	Supports concurrence	Implements complex behavior
5	Intelligent software agent [15]	Considers functionality and QOS	WS composition

TABLE 2
GENERAL CHARACTERISTICS OF THE AGENT

S.No	Agent characteristics	Description
1	Communicative	Respond to all the messages
2	Cooperative	Supports Multi-agent
3	Adoptive	Consistent in all the situation
4	Autonomy	Act independently

3 AGENT SYSTEM IMPLEMENTATION

Major advantages of the SOA is that it supports the loosely coupled application development environment, where components can be created independently called as web services. These services are described using the WSDL and the descriptions are stored in a centralized repository UDDI. The agents are providing an interface to select, invoke and compose these web services [19].

3.1 System Architecture

Web service composition is required whenever one or more services are involved in a process. The proposed model identifies the appropriate services which are take part in the composition. The following Fig 2 shows our system architecture consists of the following components performs the various tasks related to identify the dependency relation and web service selection with respect to current query (requests). Service Providers (SP) giving the web service descriptions in the form of WSDL and they are available in the centralized repository. WSDL parser component process the description of the web services and identifies the input and output parameters of the various operations [20]. Fig 3 shows parameters description in a WSDL. Service Consumers (SC) are requesting the services by giving the query through the Request Process (RP). RP identifies web services which are plays the important role in producing the composite services. The module called dependency relation responsible for getting the relationships among the services by receiving parameter information from the data store and also produces the service plan. Service plan gives the execution sequence of the operation using the plan generator. Dependency visualizer generates the graphical view of the relationships of the services.

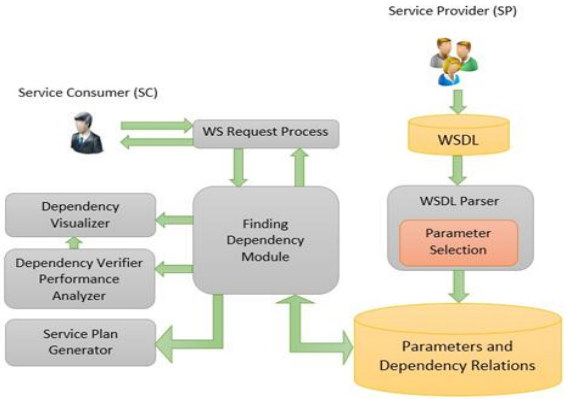

Fig 2. *System architecture*

```
<wsdl:message name="GetAllSuppliersSoapIn">
  <wsdl:part name="parameters" element="tns:GetAllSuppliers" />
</wsdl:message>
<wsdl:message name="GetAllSuppliersSoapOut">
  <wsdl:part name="parameters" element="tns:GetAllSuppliersResponse" />
</wsdl:message>
<wsdl:message name="GetMasterSuppliersSoapIn">
  <wsdl:part name="parameters" element="tns:GetMasterSuppliers" />
</wsdl:message>
<wsdl:message name="GetMasterSuppliersSoapOut">
  <wsdl:part name="parameters" element="tns:GetMasterSuppliersResponse" />
</wsdl:message>
<wsdl:message name="GetNameFromIDSoapIn">
  <wsdl:part name="parameters" element="tns:GetNameFromID" />
</wsdl:message>
<wsdl:message name="GetNameFromIDSoapOut">
  <wsdl:part name="parameters" element="tns:GetNameFromIDResponse" />
</wsdl:message>
```

***Fig 3**. Parameters description in a WSDL*

6.1 Dependency Relation Algorithm (DRA)

The concept behind the Dependency Relation Algorithm (DRA) is getting the parameters list from the repository and identifies the between the input and output parameters. To identify the dependency between the input and output parameters of service is done through the semantic comparison. There are vectors namely WS consists of input and output parameters of the web services and DR provides the dependency relation among the input and output parameters of the services. User queries processed using the following Algorithm 1 [21] [22]. The services are identified for the corresponding query and the composition sequence plan is generated to visualize the operation execution. User query is processed using the following algorithm which is depicted in the Algorithm 1 [23] [24]. The impact of DR among the services are discussed in [25]. Algorithm 2 is for dependency relation generation and is listed below. DR visualizer gives the graphical representation for the composition plan. It is achieved with following Algorithm 3 listed below

Algorithm 1: Query_Processor

```
1   Query_Processor (Qry, WS<List>)
2   Input  :  Qry – User query includes relevant and non-
                relevant informations
3   Output :  Partially identified web service list
                WS<List>
4   Begin
5       Tokenize the Qry
6       Remove the non-relevant informations
7       Prepare partially identified WS
13  End
14      Returns Classifications
15  End
```

Algorithm 2: DR_Generator

```
1   DR_Generator (WSi[], DR[][])
2   Input  :  WS[]- Web services list contains i/o
                parameters
3   Output :  DR[][]-dependency relation vector
4   Begin
5       Foreach web service i in WS[]
6         Foreach  web service j in WS[]
7           Calculate the DR
8             DR[i][j] = DR value
11        End
13      End
14      Returns DR[][] vector
15  End
```

Algorithm 3: DR_Visualizer

```
1   DR_Visualizer(DR[][], WS<List>)
2   Input  :  DR[][]- Dependency relation vector
              WS<List> Required list of web services
3   Output :  Dependency graph
4   Begin
5       Foreach relation i in DR[][]
6         Foreach web service j in WS<List>
7           Generate the graph
11        End
13      End
15  End
```

4 Results and Analysis

4.1 Case Study

To simulate the entire approach, the following case study of E-University Information scenario is considered. We assumes that this scenario consists of number of web services that provides the numerous information to the students who wants to know about the course details, university details, fees details etc. This E-University Information environment must provide the information on basis of the students' requests (Eg. Part-Time MTech Degree). The degree of composition is relies on the number of web services which are involved in the composition. Each and every relation's attribute can be treated as web services. In this scenario we have considered the following web services: WS1, WS2, WS3, WS4 and WS5 providing the following details University Information (UI), CourseName(CN), CourseType (DT), CourseFees(CF) and TotalSeats(TS) respectively. Depends upon the query the either one or more services will be returned to the users. The number of web services in the response will decide the composition is required or not and is defined by the composition plan. Table 3 shows the input and output of each service

TABLE 3

CASE STUDY - INPUT AND OUTPUT DESCRIPTION FOR E-UNIVERSITY INFORMATION WEB SERVICES

Web service id	Input	Output	Description
WS1	UI	UI, CN,CT, CF, TS	Returns course name, course type, course fee and total number of seats for the university
WS2	CN	UI,CN	Returns available courses name for all the universities
WS3	CN, CT	UI, CN,CT	Returns available courses name and its course type for all the universities
WS4	CN,CT, CF	UI, CN,CT, CF	Returns available courses name, course type and it's fees details for all the universities
WS5	UI, CN, TS	UI, CN, TS	Returns available courses name and seats available for all the universities

From the above Table 3 the dependency relation can be defined as WS5 -> WS4 -> WS3 -> WS2 ->WS1 -> none. After identifying the dependency among the web services then dependency order can be determined to prepare the composition plan. Based on the composition plan the web services are invoked and executed to produce the appropriate results.The evaluation for this model analyses the consistency between the systems generated and manually obtained composition list. We have classified the web services based on the dependency category given in the Table 4.

Definition 1: Let WS be the service set WS= {WS1, WS2, WS3, ……. WSn}.The dependency relation between two services WSi and WSj can be defined as follows WSi (op) = {O1, O2, O3,……. Om} and WSj (ip) = {I1, I2, I3,……. In}

Highly dependent $\forall x \exists y$ DR(x,y) m=n
Low dependent $\neg\forall x \exists y$ DR(x,y) m≠n
Not dependent $\neg\forall x \neg\exists y$ DR(x,y) m≠n

Where
O is output parameter
I is input parameter
m and n are number of output and
input parameters respectively
DR is dependency relation

$$DM = \begin{pmatrix} D_{00} & D_{01} & \cdots & D_{0n} \\ D_{10} & D_{11} & \cdots & D_{1n} \\ D_{20} & D_{21} & \cdots & D_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ D_{n0} & D_{n1} & \cdots & D_{nm} \end{pmatrix}$$

Where Dij ={1 55 55 5 55 555555555 55 55 5 0 55□555555

TABLE 4

DEPENDENCY CATEGORY

S.No	Value	Description
1	0	Not dependent
2	1	Low dependent
3	2	High dependent

Dependency Matrix (DM) gives the relation between the web services so as to produce web service composition. To determine the DM for the given set of web services, the parameters of the services are selected. If the output parameters of the one service is the input parameters of the other service then there is a dependency among them and is said to be fully dependent. To implement the efficient matching algorithm for the parameters the semantic approach is used. Web service composition results for the given query can be given in the Table 5 as follows

TABLE 5
USER INPUTS VS PERFORMANCE

Request	No of services in the composition	Performance %
Part-time MTech for CSE	4	83.3245
Course Fee for Part-time courses	5	80.01
Course Details for the University X	6	78.5
University details	2	92.21
MTech Course types	4	75.12

This domain specific agent model obtains the results and compared with various dimensions. The accuracy of the dependency for the existing web service calculated by DRA and compared with manually calculated values which is given in the Fig 4. Fig 5 shows efficiency of the composition.

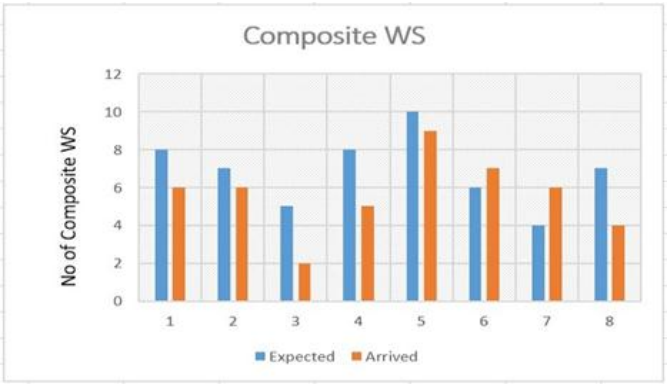

Fig 4. *WS Composition*

Fig 5. *Performance Analysis*

5 CONCLUSION

Implementation of this approach uses a web service dataset consists of 1333 WSDL descriptions and a QOS dataset for the QOS optimization. The results of this model shows that there is an improvement in the composition of web services. This model finds the dependency relation between the service parameters. The efficiency of the system is verified by giving different datasets as inputs. The results shows that the performance of the system is improved in-terms number of relevant web services in the composition into 80% accuracy. The performance of the system suffers due to the irrelevant keywords in the user's query and also the system is domain specific. By considering the multiple domain the future work can be enhanced to get better results.

REFERENCES

[1] Willmott, S., Dale, J., Burg, B., Charlton, P. and O'Brien, P. Agentcities: A Worldwide Open Agent Network. In: The Agentlink Newsletter 8, pp.13-15, 2001.

[2] WenJuan, L., YongQuan, L., Qingtian, Z.: Integrating Semantics and Agent Technology to Automatic Web Service Composition. In: IEEE 2nd Symposium on Web Society, pp. 201–206, 2010.

[3] http://www.uddi.org/pubs/uddi_v3_features.htm

[4] http://en.wikipedia.org/wiki/Universal_Description_Discovery_and_Integration

[5] http://en.wikipedia.org/wiki/Web_Services_Description_Language

[6] http://en.wikipedia.org/wiki/SOAP

[7] V. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, and Z. Zdrahal. IBROW3 - an intelligent brokering service for knowledge-component reuse on the world wide web. In Proceedings of KAW'98, 1998.

[8] Gibbins, Nicholas, Stephen Harris, and Nigel Shadbolt. "Agent-based semantic web services." Web Semantics: Science, Services and Agents on the World Wide Web 1, no. 2 (2004): 141-154.

[9] Dale, J. April Agent Platform Reference Manual. Fujitsu Laboratories of America, 2002.http://sf.us.agentcities.net/aap/

[10] Dale, Jonathan, et al. "Implementing agent-based web services." Proceedings of the AAMAS'03 workshop on challenges in open agent systems, Melbourne. 2003.

[11] Huhns, Michael N. "Software agents: The future of web services." Agent Technologies, Infrastructures, Tools, and Applications for E-Services. Springer Berlin Heidelberg, 2003. 1-18.

[12] Bian, W. U., and W. U. Xincai. "A QoS-aware method for web services discovery." Journal of Geographic Information System 2.01 (2010): 40.

[13] Maximilien, E. Michael, and Munindar P. Singh. "Multiagent system for dynamic web services selection." Proceeding of first Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE at AAMAS). 2005.

[14] Pan, ShanLiang, and QinJiao Mao. "Case Study on Web Service Composition Based on Multi-Agent System." Journal of Software 8.4 (2013): 900-907.

[15] Benaboud, Rohallah, Ramdane Maamri, and Zaïdi Sahnoun. "Semantic Web Service Discovery Based on Agents and Ontologies." International Journal of Innovation, Management and Technology 3.4 (2012): 467-472.

[16] Nikolai, Cynthia, and Gregory Madey. "Tools of the trade: A survey of various agent based modeling platforms." Journal of Artificial Societies and Social Simulation 12.2 (2009): 2.

[17] Heath, Brian, Raymond Hill, and Frank Ciarallo. "A survey of agent-based modeling practices (January 1998 to July 2008)." Journal of Artificial Societies and Social Simulation 12.4 (2009): 9.

[18] Dia, Hussein. "An agent-based approach to modelling driver route choice behaviour under the influence of real-time information." Transportation Research Part C: Emerging Technologies 10.5 (2002): 331-349.

[19] Michael Raj TF, An Integrated Approach to Rapid Automated Service Discovery of Semantic Web Service, Journal of Theoretical and Applied Information Technology, Vol. 40 No. 1, pp:78-82, JUNE 2012

[20] T.F. Michael Raj, K.S. Ravichandran,K. Rajesh, Domain Specific Web Service Composition by Parameter Classification Using Naïve Bayes Algorithm, World Applied Sciences Journal 29 (Data Mining and Soft Computing Techniques): 99-105, 2014

[21] Nguyen, Doan. "Query preprocessing: improving web search through a Vietnamese word tokenization approach." Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008.

[22] Domenig, Ruxandra, and Klaus R. Dittrich. "Query Preprocessing for Intergrated Search in Heterogeneous Data Sources." Datenbanksysteme in Büro, Technik und Wissenschaft. Springer Berlin Heidelberg, 2001.

[23] Shuying Wang, Miriam A. M. Capretz, “A Dependency Impact Analysis Model For Web Services Evolution”, IEEE International Conference on Web Services, pp: 359-365, 2009

[24] Zibin Zheng, Yilei Zhang, and Michael R. Lyu, “Distributed QoS Evaluation for Real-World Web Services,” in Proceedings of the 8th International Conference on Web Services (ICWS2010), Miami, Florida, USA, July 5-10, 2010, pp.83-90.

[25] Zibin Zheng and Michael R. Lyu, Collaborative Reliability Prediction for Service-Oriented Systems, in Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering (ICSE2010), Cape Town, South Africa, May 2-8, 2010, pp. 35-44.

[26] Fethallah, Hadjila, and Amine Chikh. "Automated retrieval of semantic web services: a matching based on conceptual indexation." Int. Arab J. Inf. Technol. 10.1 (2013): 61-66.

[27] Kavitha A, An Integrated Approach for Measuring Semantic Similarity between Words and Sentences using Web Search Engine, Int. Arab J. Inf. Technol. Yet to be bublished