Dynamic Web Service Composition Method Based on OWL-S for Educational Environment

Conference Paper · February 2018

CITATIONS
0

READS
60

2 authors, including:

Ashraf Y. A. Maghari
Islamic University of Gaza
28 PUBLICATIONS **40** CITATIONS

SEE PROFILE

Dynamic Web Service Composition Method Based on OWL-S for Educational Environment

Shady F. Samara[1], Ashraf Y. Maghari[2]

[1]Faculty of information technology, Islamic university, Gaza, Palestine, shsamara@iugaza.edu.ps
[2]Faculty of information technology, Islamic university, Gaza, Palestine, amaghari@iugaza.edu.ps

Abstract—Dynamic web service composition (DWSC) is a challenging issue that many companies and enterprises try to apply it on their services. It expands the ability of web services so that many goals can be achieved by combining little number of services. DWSC have many complex issues that make its implementation so difficult. Many algorithms have been appeared and try to solve these difficulties, but a little of them succeed to solve all difficulties of DWSC. In this paper we propose a new approach that can construct a composition between web services at run time, our approach aim to facilitate the web services work in the educational environment.
The proposed method adds semantic description to the available WSs that found in the local server, and then it clusters them and expands the consumer query words using WORDNET to facilitate the WSs discovery process. Further the suitable composition is chosen depending on the execution time of the whole composition.
We discuss many scenarios that could be occurred in the educational environment and explain how the proposed method will work, further we suggest a way to evaluate our method in the future. Hence the method may provide the scalability, compatibility and reduces the efforts that could be spent in establishing new WSs to achieve the consumer's goals.

Keywords—method, web service, clustering, education, dynamic

1.0 INTRODUCTION

Web service (WS) is any self-described software that is available over the web, which uses xml structure for description, communication and messaging to produce specific goal for the consumer. This feature for the WS make it more scalable because there are not any Restrictions in dealing with it, neither by operating systems nor programming languages. But sometimes a single service given alone cannot achieve the consumer goal, so a need for combining between many WSs appeared, this combining technique called web service composition WSC.

WSC can reduce the time and effort that could be spent in creating new services for every required goal, so instead of that we can combine between many WSs to provide new service that able to achieve new goal for the consumer.

There are two types for WSC, static and dynamic, the major differentiation between them is the time of composition construction. In the static composition the service provider must construct the composition process manually before they publish the services, but this way has a lot of limitations, the new composition still provide a specific single service that cannot adapt with the continues demand in consumers' needs, and it doesn't take into account the continues changing in the web services. So we cannot depend on this type of composition to achieve consumer needs. DWSC appeared as a solution for all problems that found in the static composition, it can combine between the web services in the run time and Take into account the demand changing in the WS without any human participation in the composition construction.

We take the educational environment such as universities, colleges and institutes as case study for our method. Usually educational enterprises have many WSs that aim to achieve many goals for its consumers, which can be classified to three groups (e.g. Students, academics and employees),

so the WSs can be clustered based on those three groups. So the DWSC is the ideal technique to reduce the time and effort in creating new services to achieve consumers goals, it can satisfy the consumers by combining between the available WSs.

In the next chapter a background about DWSC, OWL-S and educational environment will be discussed, then we will compare between many DWSC algorithms in chapter 3, the proposed method will be discussed in chapter 4, then we discuss many possible scenarios that can be occurred in chapter 5 and suggest a way to evaluate the method in chapter 6, then we conclude our work in the last chapter.

2.0BACKGROUND

2.1Dynamic web service composition

DWSC can expand the ability of the WSs by combining

Between many of them in the run time based on the required consumer's goal, it can adapt with the continues changing in requirements, goals and WSs. But there are many complexity issues in DWSC implementation, it is difficult for a normal consumer to use a web services without any particular knowledge about it, other issue about DWSC is the difficulty of choosing the suitable services that are combatable with the consumer's preferences and how to analyze his requirements to support the composition plan with the consideration of functional and non-functional attributes of the participating services.

2.2Web Ontology Language (OWL-S)

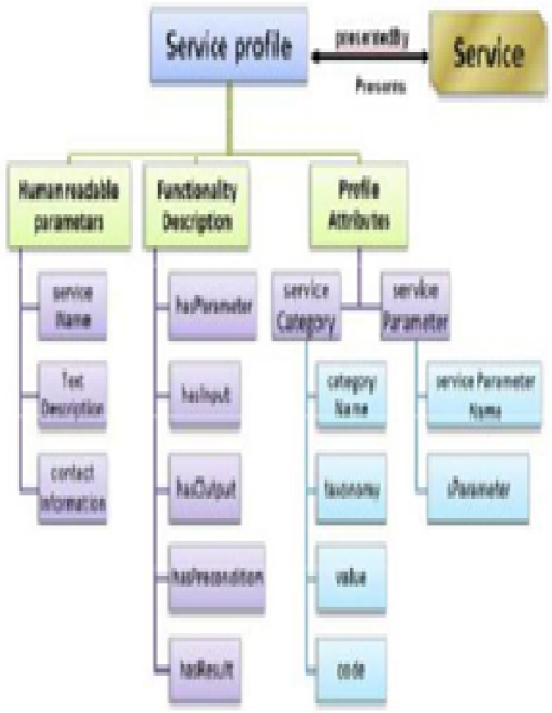
Owl is an ontology that used to describe the WS properties and capabilities, the target of owl is enable the automatic WS discovery and invocation based on the consumer query , most algorithms that rely on WSDL to find the suitable WSs for the composition depend on the keyword matching to discover them, this way have a big limitation that it can only find the services that have the same query keywords, but when we add semantic to the WSs using OWL, WSs can be discovered by matching the query keywords with its definition or meanings . OWL has three main components:

-Service profile that describe WS functionality and provide the necessary details for the WS discovery

-Service model that describe how to compose the WS and the conditions that must found to execute it.

-Service grounding that help to map WS that described by owl with WSDL.

In our paper we will use the OWL to facilities the discovery process so we care about service profile more than other components. Fig. 1shows the OWL's service profile's information [6].



2.3Educational environment

Educational enterprises like schools, universities, and institutes are one of the most important environments that need the WSs to execute the tasks and achieve their consumers goals, WSs can provide the scalability and availability for the educational environment's public due to its ability to handle with different platforms and operating systems. There are many types of consumers that can interact with the educational WSs, we can classify this users based on their goals to three groups, students, employees and academics. Every type of users

has his own WSs that can achieve his goals, so combining between available services can expand its abilities to achieve maximum number of goals with the least number of WSs.

We can benefit from applying our DWSC algorithm in the educational environments through several points:

The main problem that researchers suffer in QOS calculation is how they will determine the trusted providers that describe the correct QOS for his WSs, but in our work we will not search about trusted providers, because we will apply our work on an educational enterprise's WSs that have been implemented by a known provider.

We know the WSs provider so this facilitates the mapping process between WSDL and OWL as we will explain in the methodology section.

All WSs will be in the same repository so we don't have to determine the UDDI for the WS, this facilitate the discovery process.

3.0 RELATED WORKS

There are many algorithms that try to facilitate DWSC implementation, in this chapter we will discuss many of these algorithms and try to solve their limitations in our suggested algorithm.

EFlow is one of the most popular algorithms that depends on graph to determine the WSs flow during the run time [7], every WS is defined by its input and output, which are not enough to determine the suitable WS to participate in the composition, another limitation is that the developer must determine the processes flow at design time and every process will call the suitable service at run time, so it is semi-dynamic algorithms and it can't provide the scalability that DWSC can provide.

Another algorithm that support DWSC has been developed [8], it save the WS description into two documents, WSDL that we have mentioned before and WSCI that describes the behavior of the WS in the context of a collaborative work-flow, it perform the composition by determine the WSs that can communicate with each other to perform the process that have the input that the consumer enter and the output that consumer need, they haven't explained how their algorithm will choose the suitable composition if the algorithm return more than one composition, their algorithm has not supported QOS, it's just rely on keywords matching to discover the WSs that are suitable for the composition, the algorithm rely on WSDL to determine the input and output of each service in the workflow without determining the max number of the services that they can be combined to have a suitable composition .

A new algorithm that address the web service composition as a dependency graph is suggested in[9], the developers apply their algorithm that based on A* algorithm to find the shortest path for the composition with the minimal services, they solve a complex problem that no one as they mentioned have solved before, this problem is the redundant WSs that are invoked into the composition to achieve specific goal, it makes the graph very huge with unused services, which make the composition process more complex and increase the time for it, this problem has been noticed as a limitation in many researches[10, 11], so they describe an algorithm to eliminate the redundancy in the graph to remove all unused service, this step will reduce the time for choosing the ideal composition, their work have not taken in the consideration the QOS of the WSs, their algorithm is based on

A* that return the shortest path that link the input to the output, but sometimes the shortest path would not be the ideal solution for the composition, their algorithm depend on keyword matching to determine the WS's input/output with the request parameters.

There are many algorithms that rely on QOS to build the composition like what developed in [12], they have suggested to cluster the WSs based on its QOS to facilitate the discovery and composition process, they define the composition process as a workflow, every node in the workflow represent a specific job, then at the run time the algorithm get the suitable WS that can achieve the node job and have the best QOS, this algorithm rely on the shortest path to reach the composition output, they used manual preplanned composition workflow with nodes to invoke the

WSs automatically in the run time, so we think that this algorithm could be classified as semi dynamic web services composition .

Another algorithm that rely on QOS is mentioned in [13], they tried to solve the composition problem using WSs clustering to get the best QOS composition, they defined the WS based on its functionality and QOS, then they clustered the WSs based on the task that they can achieve. Their algorithm tries to mix between the clusters to produce the best composition that have the best QOS and can achieve the consumer goal. Further, the algorithm requires manual participate by the human to determine the WSs clusters and determine the tasks that can be performed by the cluster and the WSs order.

Another DWSC algorithm what Have been suggested in [15], its able to find the best composition based on the best response time that it take to provide the require output, it save the WSs as dependency graph that link every WS's output by the input of another WS until it link the consumer input parameter by the output that he need, it use the technique that mentioned in [9] to remove the redundancy services and reduce the time of implementation, but it still rely on keywords matching to get the WS's input/output and they have not taken other QOS parameters in the account .

Hashemian and Mavaddat have developed Another algorithm that based on Graph to construct Web Services Composition [16], they assume that a local repository that save information about existing web services is exists, this information is the service's input, output and dependency between the input and the output, they save the dependency as A graph that display the relation between the services input and output, the algorithm steps is as follow : first they convert the input to condition statement x to y, then they run BFS (Breadth First Search) algorithm on the left side of the statement to get the reachable nodes from the left side, the returned node become the left side for the next step y to z so they perform the BFS again and repeat it until they have the full composition, this algorithm solve the limitation that we talked about in the previous algorithm but it has many weakness points, it depends on WSDL for WSs description, so the participated web services will be chosen depend on keyword matching between consumer query and WS description, and if the algorithm get more than one Composition, they choose the composition randomly, so the chosen composition may not be the ideal one.

Another algorithms called QOS-WSC[17], it receives the consumer request and translate it, then it select the suitable services based on keyword matching with the translated request, it passes the request to the evaluation block that evaluate those services and pass it to the composition phase that compose it, if it doesn't found the suitable services in the local repository it will search about the service in the web then store it in the local repository with age factor to keep this service updated, this is a good algorithm but it still depend on keywords matching and this is a limitation in searching about suitable WS, and it search about the web services in the web depends on the QOS that the developer describe in his WS's WSDL, but the developer may add wrong QOS as we mentioned before, so this is another problem they didn't discuss .

FUSION is DWSC algorithm [18]; it contains 6 Subsystems as the follow: User Specification Subsystem that takes the user input and converts it to structural statement that will be the input for the next phase that called Web Services Dynamic Plan Generator. that generate a suitable plan for the composition according to the output of the previous phase, this plan will be passed to the Plan Execution Subsystem and invoke the methods instances and web service according to the generated map from the previous phase and pass it to Verification Subsystem that evaluate the result with the consumer specification to be sure that the selected services will satisfy the consumer, if the result wasn't as the consumer need then it will be passed to the next phase that called Recovery Subsystem that will roll back all process and try to search for new services to satisfy the consumer and the last phase is the User Response Generation Subsystem (URGS).

4.0EDUDC approach

In this section we propose a new dynamic web service composition's algorithm which capable to achieve educational enterprises consumer's goals by combining between WSs that saved in the local repository of the enterprise and can produce the output that required, this algorithm is based on semantic web, Figure 2 show the algorithm model that we will be discussed in this section.

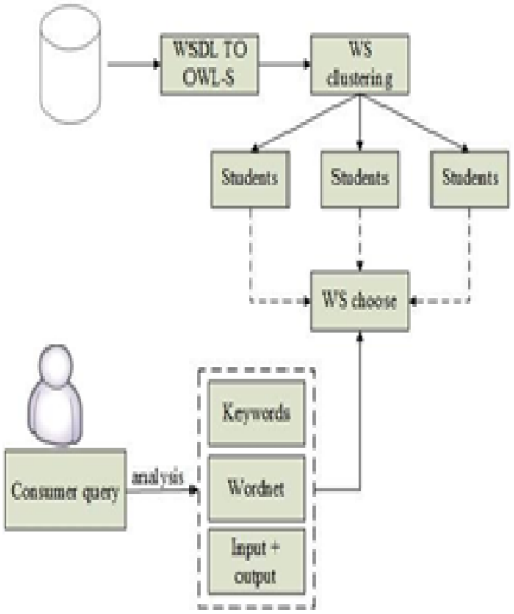


Fig. 2: EDUDC algorithm model

We can split our algorithm to five phases as the follows:

1.Mapping between WSDL and OWL-S
2.Reduce the WSs
3.Analyze the consumer requirements
4.Web services composition
5.QOS evaluation

4.1 MAPPING BETWEEN WSDL AND OWL-S

In the normal situation WSs is rely on WSDL to describe its functionality, it contain an important information such as input, output, data type, port type and other information, the WS that rely on WSDL to describe its functionality is difficult to discover because WSDL structure provide only the essential details for the WS, so we have suggested to add semantic description to the WSs in the local repository to support automatic discover and facilitate the search process.

We will use the same algorithm that [6]used to map between WSDL and OLW-S , that support the sematic description, they parse the WSDL document to extract the WS details, then their algorithm ask the WS provider that represent the educational enterprise in our paper to add the semantic details that WSDL cannot provide ,then we will cluster the semantic WSs into prepared three clusters based on consumer type, this clusters are students, academic employees, and administrative employees, clustering WSs will make the discovery more easy and faster, and adding semantic description to the WSs will facilitate the discovery process because WSs won't be rely on keyword matching that we mentioned as a weakness point in the other algorithms.

Figure 3 show the mapping process between WSDL and OWL-S according to [6], now we add the semantic description to all WSs in the repository.

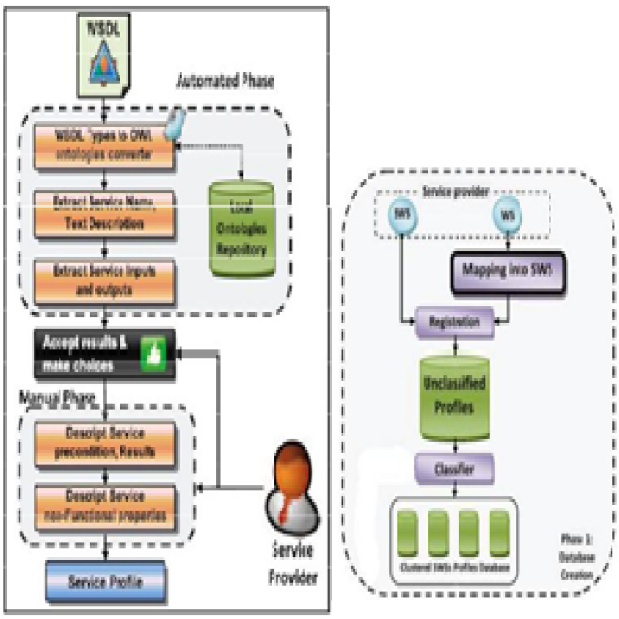


Figure3: the mapping process between WSDL and OWL-S and cluster WSs [6]

4.2 REDUCE THE WSS

There are three types of consumers in the educational environment as we mentioned before, students, academic employees, and administrative employees. Each type of this three consumers have his own goals, so to facilitate the discovery process about the suitable WSs that can be involved in the composition we will filter the WSs based on the consumer type, so if the

consumer log in as a student, only the WSs that are exist in the student cluster will be available for the discovery process, for example "marks enter" WS will not be available if the consumer login as student , the filter process will reduce the number of WSs to make the search about services more easy .

4.3ANALYZE THE CONSUMER REQUIREMENTS

we will split this phase to many sub phases, the first one is to split the consumer's query to tokens, so the complete sentence that consumer entered will be splitted to many words, so the result will be nouns, verbs, numbers and other language Structures, then we will remove any modification in the words, such as (is, was, were, are) will convert to (be), the next phase is to reduce the number of words by removing the common words such as (a, am, be, will ... etc.), and remove all words that used to write the ontology in OWL such as (WSDL,XSD ... etc.), A group of words will be produced, we will calculate the weight for each word by divide the number of times that this word found in the query to the number of all query words, and sort the words Descending order, for example if we enter a query like:" I am a Muslim, what is the month that Muslims cannot eat in it", the result will be Muslim (2/16), month (1/16), eat (1/16).

Now we will get the related words based on WorldNet, WorldNet is a database that provide short definition for a lot of English words, we can benefit from WorldNet to get the semantic meaning for the consumer query.

Now the results for consumer query analysis is a group of words and its definition so we can get all WSs from the enterprise's repository by matching the results with the WS description that the provider add in the mapping phase.

WEB SERVICES COMPOSITION

Now we have all WSs that can be involved in the composition and know the output for the composition and the input for it, the algorithm will search on the available WSs that have the same composition input, and WSs that have the same desire output for the composition, if it found any WS that have the desired output as its output and the composition input as its input, then there are no need for composition and the goal can be achieved by single WS, if the algorithm return more than one single WS that can achieve the goal, then the algorithm will select the best QOS WS, we will discuss how to calculate the QOS for the WSs later .

Else the algorithm will search for the WSs that have the desire output for the composition and put all of them in the last layer of the composition, then it search for another WSs that have the output as the same input for the previous WSs that we put.

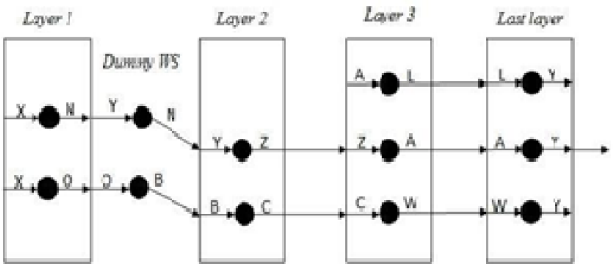


Figure4: composition establishment that have X as an input and Y as an output

4.5QOS EVALUATION

It is possible to produce more than one composition that can achieve consumer's goals, in this case we suggest choosing the fitness composition based in its QOS.

QOS is the non-functionality attributes of the WS like the cost, execution time, security, availability and other attributes, the main problem that researchers suffer in QOS calculation is how will they determine the trusted providers that describe the correct QOS for his WSs, most of providers provide an incorrect QOS values to motivate the consumer to use their service, but in our work we will not search about trusted providers , because we will apply our work on specific enterprise WSs that have been implemented by known provider, we can get each WS QOS from the SLA document that the provider can publish many details about the WS in it, and it can be as a contact between the consumer and the provider .

The local consumers in the educational environment will not be interested with price attribute because all of the WSs that exist in the repository are usually free, and the availability

will be the same for all WSs because they are found in the same repository, we assume that the most factor that important for the consumers is the execution time of the composition.

There are many techniques to combine WSs to achieve specific goal, WSs can be combined in parallel or in sequence Figure 5 show the difference between them.

Figure5: sequence composition and parallel composition

In the parallel we can calculate the total execution time for the combined WSs by get the min value from the services, while the sequence composition could be calculated by summation of all combined services in this composition.

So after the calculation the best composition will be the one with the less time to execute.

4.6SCENARIOS

First of all we have to add the semantic description to all WSs in the educational enterprise's repository based on the steps that we mentioned in Section 6.1. Figure 6 shows an example for WSs that could be available in the educational environment repository; we will not discuss the semantic description for every WS in details here

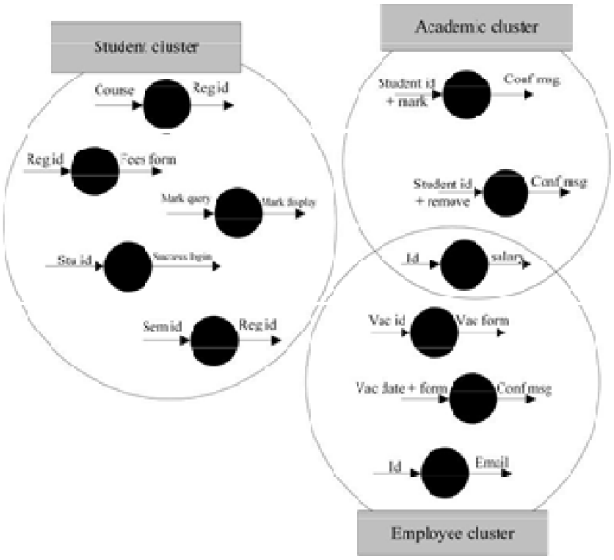


Figure 6: clustered web services

Suppose that a student want to register new course called "algorithms", the input query of the process is "I want to register algorithms course" and the output is a successes registration message with course fees details, there are many WSs in the student cluster as shown in Figure 6, but there are no WS can achieve the process goal alone, so we must combine between two services to achieve student goal, the composition algorithm will be applied as the following steps:

1- Student query will be analyzed based on the steps that mentioned in section 6.3, so the result will be: "register, algorithms, course ", then the algorithm will get the definitions for this words from WorldNet to expand the search process and facilitate the discovery process.

2- The algorithm will search in the student cluster about any WS that have register, algorithms, course or their

Definitions that have been returned from WorldNet in its semantic description and get all matching results.

3- The algorithm will combine between services based on the technique that explained in section 6.4, Figure 7 show the possible composition that can achieve the student goal, if more than one composition can be as a solution for the student problem then the algorithm will choose the composition with the minimum execution time

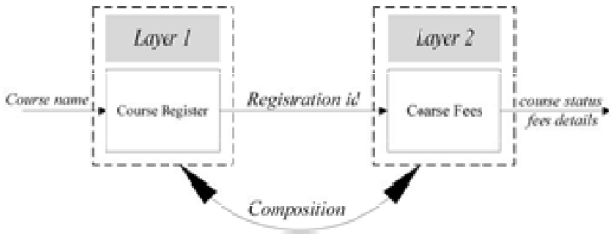


Figure 7: Course registration composition

Another scenario for our algorithm when an administrative employee want to ask for vacation, he enter "I want to ask for normal vacation in 25-12-2016". The input of the process is the vacation id and date and the output is a success confirmation message, to achieve this goal we have to combine between two WSs as shown in Figure 7 .the composition algorithm will be applied as the following steps: 1- The employee query will be analyzed based on the steps

That mentioned in section 6.3, so the result will be: "vacation, request, 25-12-2016 ", then the algorithm will get the definitions for this words from WorldNet to expand the search process and facilitate the discovery process.

2- The algorithm will search in the employee cluster about any WS that contains vacation request or date in its semantic description and get all matching results.

3- The algorithm will combine between services based on the technique that explained in section 6.4, Figure 8 show the possible composition that can achieve the student goal, if more than one composition can be as a solution for the student problem then the algorithm will choose the composition with the minimum execution time

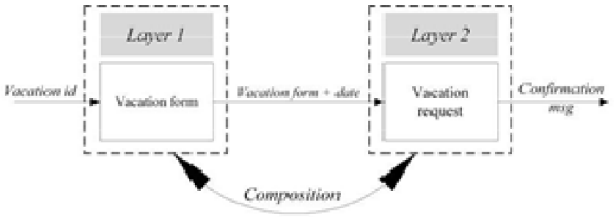


Figure 8: vacation request composition

Another scenario for our algorithm when an academic employee want to enter mark for specific student, he enter "put 95 mark for 20150197". The input of the process is the mark and the student id and the output is a success confirmation message, to achieve this goal there are no need for composition because we have a WS in the repository that can achieve this goal alone as shown in Figure 9.

Figure 9: enter mark web service

5.0FUTURE WORK

In our work we calculate the total QOS for the whole composition according to the execution time off all participated WSs in the composition, we think that this is enough to choose the suitable composition, but we have not enough time to evaluate this assumption, and let the evaluation for the future.

We will evaluate our approach by implementing a group of WSs that could be available in the educational environment and try to ask for specific task that no one of the existing services can achieved a lone, we will compare between normal system and our system based on many factors such as:

The ability of achieve the consumer goal, the execution time.

How many solution of the problem that can be got from every system.

6.0 CONCLUSION

In this paper we suggested a new algorithm that can handle with the complexity issues of the dynamic composition , our algorithm take the educational environment as a case study to apply our algorithm on it, it rely on OWL-S to add semantic description to the web services and cluster them based on the consumer type that can use this service, we add some additional features to our algorithm such as its ability to analyze the consumer query and QOS calculation, then we mention one scenario for every consumer type and explain how we can evaluate this algorithm

REFERENCES

[1] Y. R. E. Pejman, P. M. Esfahani, and A. Salajegheh, "Web service composition algorithms : A survey,"
Proceedings of the International MultiConference of Engineers and Computer Scientists, 2012.

[2] M. C. Jaeger and H. Ladner, "Improving the QoS of WS compositions based on redundant services," in International Conference on Next Generation Web Services Practices (NWeSP'05), 2005, p. 6 pp.

[3] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," IEEE Transactions on software engineering, vol. 30, pp. 311-327, 2004.

[4] C. Zhang, S. Su, and J. Chen, "A novel genetic algorithm for qos-aware web services selection," in Data Engineering Issues in E-Commerce and Services, ed: Springer, 2006, pp. 224-235.

[5] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, et al., "OWL-S: Semantic markup for web services," W3C member submission, vol. 22, pp. 2007-04, 2004.
[6] T. A. Farrag, A. I. Saleh, and H. A. Ali, "Toward SWSs discovery: mapping from wsdl to owl-s based on ontology search and standardization engine," Knowledge and Data Engineering, IEEE Transactions on, vol. 25, pp. 1135-1147, 2013.
[7] S. I. F. Casati, and L. Jin, "Adaptive and dynamic
service composition in
EFlow," Springer Verlag., 2000.
[8] P. P. Chan and M. R. Lyu, "Dynamic web service composition: A new approach in building reliable web service," in Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on, 2008, pp. 20-25.
[9] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Automatic web service composition with a heuristic-based search algorithm," in Web Services (ICWS), 2011 IEEE International Conference on, 2011, pp.81-88.
[10] M. M. Shiaa, J. O. Fladmark, and B. Thiell, "An incremental graph-based approach to automatic
service composition," in Services Computing, 2008. SCC'08. IEEE International Conference on, 2008,
pp. 397-404.
[11] S. Kona, A. Bansal, M. B. Blake, and G. Gupta, "Generalized semantics-based service composition,"
in Web Services, 2008. ICWS'08. IEEE International Conference on, 2008, pp. 219-227.
[12] A. K. Tripathy, M. R. Patra, M. A. Khan, H. Fatima, and P. Swain, "Dynamic web service composition with qos clustering," in 2014 IEEE International Conference on Web Services (ICWS), 2014, pp. 678-679.
[13] Z. Xiang, S. Deng, and H. Gao, "Service Selection Using Service Clusters," in Services Computing (SCC), 2015 IEEE International Conference on, 2015, pp. 769-772.
[14] S. K. Bansal, A. Bansal, and M. B. Blake, "Trust-based dynamic web service composition using social network analysis," in Business Applications of Social Network Analysis (BASNA), 2010 IEEE International Workshop on, 2010, pp. 1-8.
[15] Y.-M. Xia and Y.-B. Yang, "Web service composition integrating qos optimization and redundancy removal," in Web Services (ICWS), 2013 IEEE 20th International Conference on, 2013, pp. 203-210.
[16] S. V. Hashemian and F. Mavaddat, "A graph-based approach to web services composition," in The 2005 Symposium on Applications and the Internet, 2005, pp. 183-189.
[17] M. Y. J. Farhan Hassan Khan, Saba Bashir,Aihab
Khan, Malik Sikandar and Hayat Khiyal, "QoS Based Dynamic Web Services Composition & Execution," International Journal of Computer Science and Information Security.
[18] C. o. C. Debra VanderMeer , Anindya Datta , Kaushik Dutta , Helen Thomas , Krithi Ramamritham and Shamkant B. Navathe, "FUSION: A System Allowing Dynamic Web Service Composition and Automatic Execution," EEE International Conference on E-Commerce, 2003. CEC 2003., 2005.
[19] P. P. C. a. M. R. Lyu, "Dynamic web service composition: A new approach in building reliable
web service," Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on, 2008, pp. 20-25.