



Solving the Problem of Uncertainty Execution Effect Based on Web Service Composition

R.Rajesh, S.Pravisha

Assistant Professor, Student

Department of Computer Science and Engineering, IFET College of Engineering, Villupuram, India

*Abstract- The web service composition is a graph search problem from the point of semantic input-output message structure matching, i.e., it do not take into consideration the nonfunctional properties (NFPs). The proposed graph search-based algorithm can be applied to the general service composition problems. Different from the previous methods, it can generate all the feasible solutions according to a user's request. This method is able to calculate, given a request, an extended service dependency graph which represents a suboptimal solution is dynamically generated for the request. Then the solution is improved using a backward heuristic search algorithm, based on the well-known A*algorithm, which finds all optimal solutions from the point of view of the number of services and execution path (runpath). Thus, it maximizes the parallel execution of services and minimizes the number of services. To define set of optimizations and to reduce the graph size, based on the redundancy analysis and service dominance. I include a method to reduce dynamically the possible paths to explore during the search by filtering equivalent compositions. . Moreover, in order to improve the scalability of our approach, a set of dynamic optimization techniques have been included.*

Keywords:- web service composition, A algorithm, graph size, dynamic optimization*

I. INTRODUCTION

Web service composition is the process of combining and reusing the existing web service to create a new business process according to user requirements. In the existing methods for automatic service composition, the requirements of users mainly include the available inputs that they can provide, the outputs that they expect, and the quality of service (QoS) standards that they require. These requirements, including available inputs, required outputs, QoS standards, and preferences, are all provided explicitly by users at the beginning, and the composition of services is

guided by this information. In fact, some other user-related information can also greatly affect the composition of services, but often is not given explicitly by users at the beginning. This information is mainly some context conditions, which are often related to users and implicit in the service composition scenarios.

II. RELATED WORK

As a matter of fact, the fundamental reason of such situations is that many available services have applicable conditions or use restrictions, i.e., they are not universally applicable. Most of the available services in our real world are actually like this, and some constraints are often imposed on them by their providers, such as their effective time and available areas. These constraints specify the context conditions that must be met to ensure the correct execution of the service or the proper interaction with it. In this situation, even if the given inputs can match its data interface, a service may still be unable to execute correctly. Therefore, only matching parameters between service interfaces is not enough, and more factors should be given full consideration when we are dealing with service composition. However, most of the existing methods are basically based on input and output interfaces of services. For them, services are merely represented by their input and output parameters, and service compositions are implemented through their matching. Thus, they must be improved.

we concentrate our attention on the topic of composing services with constraints. First, through some real world business scenarios, we describe the problem. Then, service constraints and formal expressions are defined. We divide the elements of a web service into two parts: service intension and service extension. Finally, a graph search-based algorithm and two different preprocessing methods are presented to advance the field.

The contributions of our work are as follows.

1. Service constraints are presented and defined for a web service, which specify the conditions that must be met to ensure the correct execution of a service.
2. The solution can solve the problem of automatic service composition while considering service constraints, which have gone beyond what the existing techniques can handle.
3. The graph search-based algorithm can be applied to the general service composition problems. Different from the previous methods, it can generate all the feasible solutions according to a user's request.

III. PROPOSED METHOD

The proposed graph search-based algorithm can be applied to the general service composition problems. Different from the previous methods, it can generate all the feasible solutions according to a user's request. The method is able to calculate, given a request, an extended service dependency graph which represents a valid but suboptimal solution for the request. The heuristic search algorithm, based on the well-known A*, finds all optimal solutions from the point of view of the number of services and execution path (runpath). This, it maximizes the parallel execution of services and minimizes the number of services. We define set of optimizations to reduce the graph size, based on the redundancy analysis and service dominance. We include a method to reduce dynamically the possible paths to explore during the search by filtering equivalent compositions.

- It can generate all the feasible solutions according to a user's request.
- To improve the reusability and simplify the application logic, atomic web services are often simple.

Thus the ability to efficiently and effectively select and integrate inter organizational and heterogeneous services is important toward the development of web service applications.

IV. Modules in Proposed Work

Web service composition

In order to compose web services, we must define the relationship among services. From a functional point of view, a web service is a software component that receives a set of inputs and generates a set of outputs after the execution. Outputs from a service can be provided as inputs to other service only if there is a semantic relationship between them. In our approach, we have modeled this restriction as a hierarchical class/subclass relationship between concepts, so we consider that an output of a service matches the input of other service *Isi* when *Oso* is a subclass of *Isi*. This workflow, therefore, has services and a set of control structures that define both the behavior of the execution flow and the inputs/outputs of the services related to those structures.

Web service Graph

Web services composition requires the combination of many atomic services that can be executed in sequence or in parallel as previously mentioned. Given a service request, an extended service dependency graph with a subset of the original services from an external repository is dynamically generated. This subset contains the solutions that meet the request and consists of a set of layered services (splits) connected in sequence. Each layer contains all services from the repository that can be executed with the outputs of the previous one. In order to speed up the calculation of the graph, we used a pre-computed table that maps each input to the services that use it. Thus, for each output generated in a layer, we can obtain all possible services for the next layer very quickly.

Service Optimization:

In order to achieve a significant performance improvement on the search process, we designed two techniques that reduce the number of possible paths to explore:

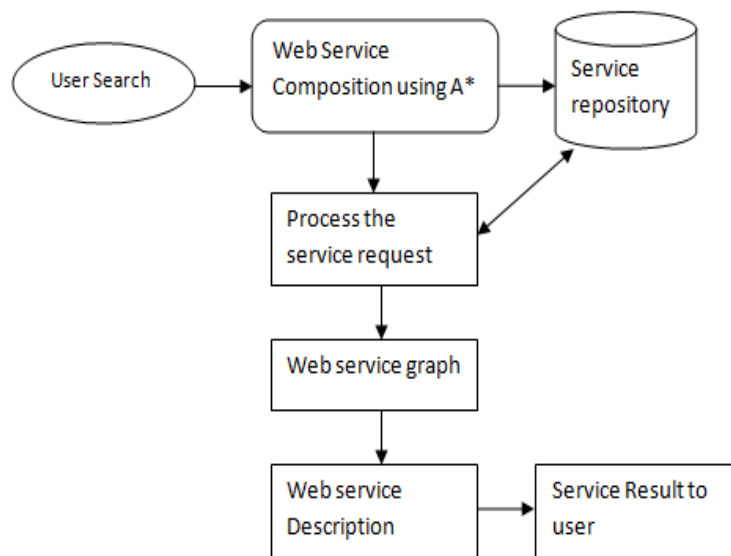
Offline Service Compression

The essence of this technique is to replace equivalent services from each layer in the graph by the representative service, which implies a lower number of paths to explore during the search.

Online Node Reduction

This technique consists in the combination of equivalent neighbors during the A* search process. Given that a node can generate equivalent neighbors (different combination of services that together are equivalent) a mechanism to delete this type of redundancy must be implemented

Architecture of proposed work

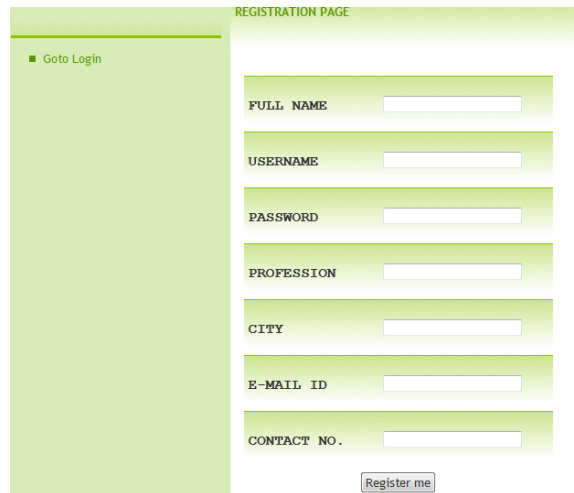


V. EXPERIMENTAL RESULTS

Web Service means Collection of web pages that makes service to customers according to request who can register a particular service. Here we have maintained a composition of web services through a single application. So that many phases are available for single service and applicable for composition. The registered users can send query as request and response from a web services. The service has been created and generated by server. The user search through multi services within a single service. Here composition applicable for any websites and search within online credit card transaction. To check whether the books are avail or not and search through server and response acknowledgement. The result provides efficient search to users from a multi services.

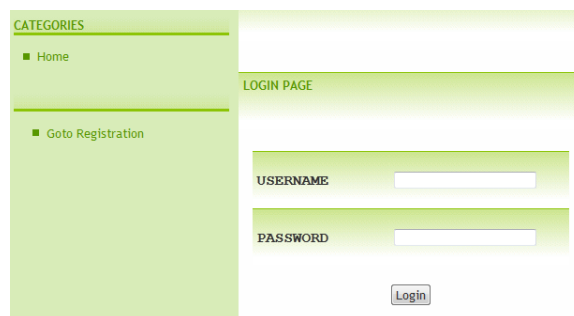
Here the existing system defines user search goals as the information on different aspects of a query that user groups want to obtain. Information need is a user's particular desire to obtain information to satisfy his/her need. User search goals can be considered as the clusters of information needs for a query. The inference and analysis of user search goals can have a lot of advantages in improving search engine relevance and user experience. There are many issues in the systems. They are, what users care about varies a lot for different queries, finding suitable predefined search goal classes is very difficult and impractical. Analysing the clicked URLs directly from user click-through logs to organize search results.

Registration



The screenshot shows a registration form titled "REGISTRATION PAGE". On the left, there is a sidebar with a "Goto Login" link. The main form area contains several input fields: "FULL NAME", "USERNAME", "PASSWORD", "PROFESSION", "CITY", "E-MAIL ID", and "CONTACT NO.". A "Register me" button is located at the bottom of the form.

Login



The screenshot shows a login form titled "LOGIN PAGE". On the left, there is a sidebar with "Home" and "Goto Registration" links. The main form area contains two input fields: "USERNAME" and "PASSWORD". A "Login" button is located at the bottom of the form.

Service composition page

TYPE

COST

SECURITY

TIMING

QUALITY

PRODUCT NAME

PRODUCT DESCRIPTION

PRODUCT COST

PRODUCT IMAGE

SERVICE COMPOSITION PAGE

TYPE

COST

SECURITY

TIMING

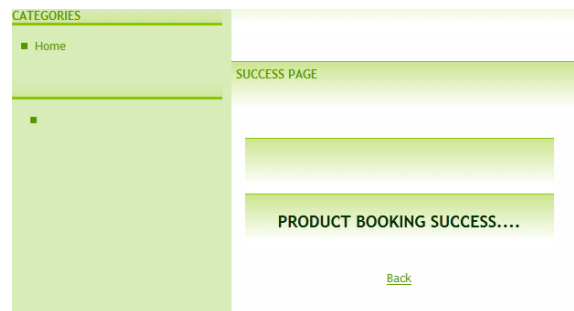
QUALITY

Product details

Suggestions			
Product Id	Product Name	Product Description	Product Cost
1	gjhj	srgfrhf	230 <input type="button" value="Buy"/>
3	sss	aaaa	300 <input type="button" value="Buy"/>
4	sss	sss	124 <input type="button" value="Buy"/>
5	sss	tyty	300 <input type="button" value="Buy"/>

User suggestions

User Suggestions				
Username	Product Name	Product Description	Product Cost	Time
selva	Nokia X	First android in Nokia	5600	100
selva	sss	aaaa	300	439
pravisha	sss	aaaa	300	73



VI. Algorithm

1. Once the graph is calculated, a search over it must be performed.
2. The search algorithm will traverse the graph backwards, from the solution (the service whose inputs are the outputs wanted by the requester), to the initial node (the service whose outputs are the provided inputs).
3. our heuristic algorithm is based on an implementation of the A* heuristic search.
4. There are three principal concepts in this type of algorithms: the neighborhood function, the cost function and the heuristic function. Calculate, for each input of a node, a list of services from the previous layer that provide it.
5. If there are no services in the previous layer for that input, a dummy service that generates this input and receives the same input is created.
6. This dummy holds the dependency so it can be resolved later. Make all combinations among services from each list.
7. These combinations will generate all possible neighbours from the current node. Remove all equivalent neighbors

VII. Conclusion

Service constraints are used to ensure the correct execution of the service or proper interaction with other services, thus having a significant impact on service composition. However, they were not put into account in the previous work. This work deals with automated service composition while considering them. Through some real-world business scenarios, we show that some available services in a specific business scenario have the same inputs and outputs, and can perform the same task, but have different constraints.

Given a request, a service dependency graph which represents a suboptimal solution is dynamically generated. Then, the solution is improved using a backward heuristic search based on the A* algorithm which finds all the possible solutions with different number of services and runpath. Extensive experiments are conducted via a publicly available test set, and experimental results show the effectiveness and admissibility of our approach. This can help us perform more extensive experiments for the proposed methods, and also provide support for other future work that considers such factors and integrate with other research outcomes. In addition, QoS information is planned to be used in order to directly get rid of the services and paths that do not meet the QoS standards, thereby reducing the search space and improving the efficiency of the proposed algorithm.

REFERENCES

1. D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, Jun. 2007.
2. M. Beek, A. Bucchiarone, and S. Gnesi, "A survey on service composition approaches: From industrial standards to formal methods," IEEE Comput. Soc. Press, New York, NY, USA, Tech. Rep. 2006TR-15, Istituto, 2006, pp. 15–20.
3. D. Benslimane, Z. Maamar, Y. Taher, M. Lahkim, M. C. Fauvet, and M. Mrissa, "A multilayer and multiperspective approach to compose web services," in *Proc. AINA*, May 2007, pp. 31–37.
4. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, "Automatic composition of e-services that export their behavior," in *Proc. ICSOC*, Dec. 2003, pp. 43–58.
5. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella, "Synthesis of underspecified composite e-services based on automated reasoning," in *Proc. ICSOC*, Nov. 2004, pp. 105–114.
6. A. Brogi, S. Corfini, and R. Popescu, "Composition-oriented service discovery," in *Proc. SC*, Apr. 2005, pp. 15–30.
7. A. Brogi and S. Corfini, "Behaviour-aware discovery of web service compositions," *Int. J. Web Serv. Res.*, vol. 4, no. 3, pp. 1–25, Jul.–Sep.2007.
8. A. Brogi, S. Corfini, and R. Popescu, "Semantics-based compositionoriented discovery of web services," *ACM Trans. Internet Technol.*, vol. 8, no. 4, article 19, Sep. 2008.
9. Z. J. Ding, J. L. Wang, and C. J. Jiang, "An approach for synthesis Petri nets for modeling and verifying composite web services," *J. Inf. Sci. Eng.*, vol. 24, no. 5, pp. 1309–1328, Sep. 2008.
10. W. Kongdenfha, H. R. Motahari-Nezhad, B. Benatallah, F. Casati, and R. Saint-Paul, "Mismatch patterns and adaptation aspects: A foundation for rapid development of web service adapters," *IEEE Trans. Serv. Comput.*, vol. 2, no. 2, pp. 94–107, Apr.–Jun. 2007.