International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.2, 2015

REVIEW ARTICLE

ISSN: 2321-7758

# ACO FOR COMPOSITE WEB SERVICES USING OWL-DL AND RULES

**MEENA.S[1], SASIKALA.R[2], KEERTHIKA.V[3]**
[1,2] UG Scholar, [3]Asst. Professor, Department of Computer Science and Engineering
Veerammal College of Engineering, Dindigul

Article Received: 18/03/2015

Article Revised on:31/03/2015

Article Accepted on:03/04/2015

**MEENA.S**

**SASIKALA.R**

**ABSTRACT**

Based on the studies of semi automated composite we found that, Web services select concrete services based on the functional and/or non- functional attributes. They do not consider relationships between these attributes in the description of services or the user constraints. Thus, we propose an approach, which relates services to objects (resources) maintained by these services. The user can impose his constraints on the objects affected by the requested services. The affected object and their relationships are described in an intermediate ontology using OWL-DL and SWRL languages. Our selection strategy considers the relationships between services by looking for the dependent instances (conforming objects values) of affected objects that satisfy the user constraints and by combining the related services to get conforming composite services. The proposed selection approach of conforming composite services is implemented and integrated using Ant Colony optimization follow paths with conforming values of concrete services using SPARQL query.

Keywords: Concrete service, OWL, SPARQL query, Ant Colony.

## 1. INTRODUCTION

Web services composition is a highly active studied research direction. It provides mechanism to aggregate a multiple services into one composite service. In contrast to the traditional Web service composition based on IA planning or workflow, there are a lot of Web services offered by different providers providing the same functionality; however, these services that have different values of attributes describing functional or non-functional properties (NFP) could be gathered into a collection of Web services (category, task, type, community, abstract service) and used to select and to deter-mine the most appropriate concrete (instance) service.

The ultimate goal of the composition research is to provide a fully automatic process. Such semiautomatic Web service compositions are essentials when for example considering user constraints for each abstract service where specific Web services are not predefined. All semiautomatic approaches are based on an existing abstract composition and on instantiating this abstract composition by selecting the most appropriate concrete Web services with respect to user constraints and service properties. Selection for the purpose of the composition process depends on user constraints imposed on functional properties (FP) (for example, the user seeks a three stars hotel) or NFP(for example, the user seeks a service with high availability).It has to take care of the difference between conforming composite Web service when we talk about constraints on the Fan the optimized composite service when we impose constraints on NFP. Most of the solutions that calculate the optimized composite service are generally syntax-based approaches, focusing on the optimization problem and use some optimization algorithms such as Integer programming or genetic algorithms. When we consider the problem of con-forming composite services, semantic-based approaches are generally used to locate and match functional attributes describing individual concrete services. However, these approaches focus on describing and locating single services. Thus, they cannot capture relationships between services that are essential information to determine their compatibility and their dependency when we select concrete services for the purpose of the composition.

In our work, we focus on the objects (resources) maintained by the concrete services and their semantic relationships described in an independent ontology. Therefore, services are indexed according to these affected objects; and user constraints are imposed on the affected objects. Our selection strategy considers the relationships between services when it takes into account the relationships between their related objects. To relate services to their affected objects, we propose a semantic framework called abstract services ontology that clusters concrete services into different abstract services according to their functional characteristics and the type of manipulated objects. Based on imposing constraints on the affected objects, our selection approach generates dynamically SPARQL queries to infer dependent instances of affected objects (conforming objects values) and related ser-vices that will be combined in order to form conforming composite services.

In this paper, we present a semantic selection approach of conforming composite Web services. Our contributions are threefold. First, we propose an ontology framework that allows bundling concrete services to their abstract services according to their functionality and their affected functional objects. An intermediate independent ontology has been pro-posed in order to provide a shared and common description of affected objects and their relationships. Second, we show how the user can draw a new kind of constraints, semantic constraint by considering the relationships between objects. Third, we propose and implement a semantic approach for the selection using semantic Web tools to locate conforming composite concrete services.

## 2.RELATED WORK

In this section, we present different related works that are based on an existing abstract composition and on instantiating this abstract composition by selecting the most appropriate concrete Web services with respect to the user constraints. A classification of different related works has been made according to some important criteria that have to be considered when comparing service selection for the composition of Web services. In this work, we focus on the most important requirements and we will show the very closest works according to these requirements.

### 2.1Requirement1: expressive service description

Atomic or composite Web services are typically described by using languages like BPEL or OWL-S. which provide mechanisms for the description of Web service com-position. However, the process of Web service composition is intended to be static in the sense that the composition flow is generated off-line. Unfortunately, the main impediment to current Web service standards is their limited support to dynamic automated composition and selection.

### 2.2 Requirement 2: expressive user constraints

Services are described by their FP (input, output, pre-condition, and affect), functional attributes, and their NFP. User can impose constraints on FP, functional attributes, and/or NFP. He can also impose semantic constraints by specifying semantic relationships between services attributes.

Most existing work of service composition and selection are based on NFP. Zeng et al. address the issue of selecting Web services for the purpose of their composition in a way that maximizes user satisfaction expressed as utility functions over quality-of-service (QoS) attributes, while satisfying the constraints set imposed by the user and by the structure of the composite service. Two selection approaches were described and compared: one based on local selection of services and the other based on global optimization using integer programming. The set of user and composition constraints defined in was the background for other studies using other models of optimization like mixed integer programming , genetic algorithm, Ant colony optimization , and constraint programming.

**2.3 Requirement 3**: efficient selection strategy

Define selection strategy that takes into consideration services description, their interdependencies, and different user constraints.

For selection mechanism, authors use many strategies in instantiating abstract services by concrete services in order to find the best solution that meets all criteria defined by the user according to services properties. We distinguish between global solvers and local solvers. Global solvers are well suited for global constraints. When authors use local solvers, they also define backtracking and propagation process to find alternatives to some previously chosen node instantiation solution. Once the backtracking process execution terminates, the composition solution must be recomputed and may require the propagation of the newly chosen solution to the higher level in the hierarchical order of the composition flow. Monfroy et al. and Zahoor et all involve backtracking when selection process needs to choose dependency (conforming) services described by specifying added properties called (Work With, link, etc.). The class of composite services solutions that use a backtrack search, is known as NP-hard .

**3. Global architecture for service selection and composition**

A typical architecture of our system is shown in Fig.1.Itinvolves four types of players: service providers, information agents, ontology provider, and the composer. Serviceproviders offer the atomic concrete services. The information agents crawl the Web to collect information about concrete services from the atomic providers and index them into a meta-ontology called abstract services ontology according to the specialized functionalities (abstract services). The vocabulary shared by these abstract services called inter-mediate ontology is defined by the ontology provider. The ontology provider has also to define the offered functionalities (abstract services) and the information agent assigned to each abstract service.

Each information agent is responsible for the automatic discovery at runtime/off time of the concrete services that fulfill the corresponding abstract service functionality. Information agents gather information about concrete services such as cost, price, type, and affected objects. The obtained information is saved in the abstract services ontology and used by the composer to find the most suitable concrete services according to the intermediate ontology.

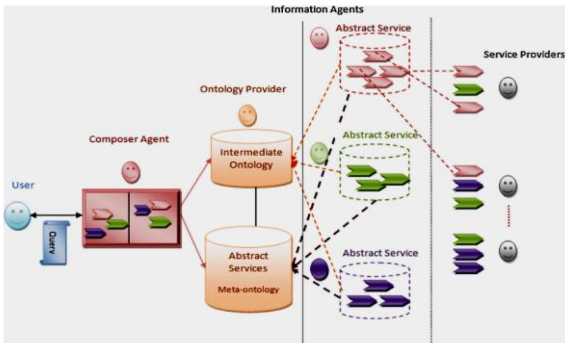

**Fig. 1 Global architecture for service selection and selection**

**4.ACO ALGORITHM**

The ACO algorithm has been demonstrated to be an effective means of solving complex combinatorial optimization problems. In ACO, the positive feedback of pheromone deposits on arcs comprising more optimal node arc tours (paths), allows the next cycle (iteration) to progress toward an optimal solution [21][22]. ACO mimics the behavior of foraging ants. Ants deposit pheromones on the paths that they move along. The pheromone level deposited on a particular path increases with the number of ants passing along it. Ants adopt pheromones to communicate and cooperate with each another in order to identify the shortest paths to a destination. ACO is applied to the TSP first, since it enables an efficient evolution toward quality sub/optimal.

An ant is a simple computational agent in the ant colony optimization algorithm. It iteratively constructs a solution for the problem at hand. The intermediate solutions are referred to as solution states. At each iteration of the algorithm, each ant moves from a state $x$to state $y$, corresponding to a more complete intermediate solution. Thus, each ant $k$computes a set $A_k(x)$of feasible expansions to its current state in each iteration, and moves to one of thesein probability. For ant $k$, the probability $p_{xy}^{k}$of moving from state $x$to state $y$depends on the combination of two values, viz., the *attractiveness* $\eta_{xy}$of the move, as computed by some heuristic indicating the *a priori* desirability of that move and the *trail level* $\tau_{xy}$of the move, indicating how proficient it has been in the past to make that particular move.

The *trail level* represents a posteriori indication of the desirability of that move. Trails are updated usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

In general, the $k$th ant moves from state $x$to state $y$with probability

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{y\in \mathrm{allowed}_y}(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

The amount of pheromone deposited for transition from state $x$to $y$, 0 ≤ $\alpha$is a parameter to control the influence of $\tau_{xy}$, $\eta_{xy}$is the desirability of state transition $xy$(*a priori* knowledge, typically $1/d_{xy}$, where $d$is the distance) and $\beta$≥ 1 is a parameter to control the influence of $\eta_{xy}$. $\tau_{xy}$and $\eta_{xy}$represent the attractiveness and trail level for the other possible state transitions.

## 5. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a semantic approach to select conforming composite Web services based on imposing constraints on the affected objects of the requested services. Our selection for the purpose of the composition locates dependent services by considering the relationships between affected objects described in an intermediate ontology. Our selection implementation generates dynamically SPARQL queries to obtain conforming objects values and related concrete services.

The main characteristics of our approach are the facts that we neglect non-conforming values and we generate a polynomial number of SPARQL queries to reduce execution time. Imposing constraints on the affected objects reduces significantly the number of conforming objects values and so forth the number of generated composite services.

## REFERENCE

[1]. Zeng L, Benatallah B, Ngu A, Dumas M, Kalagnanam J, Chang H (2004) Qos-aware middle-ware for services composition. IEEE Trans Softw Eng 30(5):311–327

[2]. Hassine AB, Matsubara S, Ishida T (2006) A constraint-based approach to horizontal web service composition. In: International semantic Web conference, pp 130–143

[3]. Monfroy E, Perrin O, Ringeissen C (2008) Dynamic web services provisioning with constraints. In: Proceedings of 16th international conference on cooperative information systems, CooPIS'08, OTM conferences, Lecture notes in computer science, vol 5331. Springer, Berlin, pp 26–43

[4]. Monfroy E, Perrin O, Ringeissen C (2008) Modeling web ser-vices composition with constraints. In: Selected papers of the third Colombian conference on computer science, special issue of Revista Avances en Sistemas e Informtica 5(1)

[5]. Zahoor E, Perrin O, Godart C (2009) Rule-based semi automatic web services composition. In: Proceedings of the 2009 congress on services—I (SERVICES 09). IEEE Computer Society, Washington, DC, pp 805–812.

[6]. Sirin E, Parsia B, Wu D, Hendler JA, Nau DS (2004) HTN planning for web service composition using SHOP2. J Web Semant 1(4):377–396

[7]. Gamha Y, Bennacer N, Vidal-Naquet G, El Ayeb B, Romdhane LB (2008) A framework for the semantic composition of web services handling user constraints. In: ICWS 2008, pp 228–237

[8]. Yu HQ, Reiff-Marganiec S (2009) A backwards composition con-text based service selection approach for service composition. In: SCC, IEEE international conference on services computing, pp 419–426

[9]. Gooneratne N, Tari Z, Harland J (2007) Matching strictly depen-dent global constraints for composite web services. In: Proceedings of the fifth European conference on Web services (ECOWS '07). IEEE Computer Society, Washington,DC,pp139–148.