Web Services Composition Handling User Constraints: Towards a Semantic Approach

Amel Boustil
Computer Science Department, Mentouri University
LIRE Laboratory, Constantine, Algeria
Amel.boustil@lip6.fr

Nicolas Sabouret
Laboratoire Informatique de Paris 6, 4, Place Jussieu 75005, Paris, France
Nicolas.sabouret@lip6.fr

Ramdane Maamri
Computer Science Department, Mentouri University
LIRE Laboratory, Constantine, Algeria
rmaamri@yahoo.fr

ABSTRACT
Service composition is a very complex and challenging task, especially when it has to consider not only the services properties and their dependencies but also users constraints. In this paper we propose a semantic based approach for web service composition, which allows users to define different kinds of constraints: local, global and choreography constraints. Services are organized in a framework capturing dependencies between abstract and concrete services. We also provide a survey of the state of the art of the field and we make some suggestions as to where the research in this area might be heading and how our future work need to be addressed.

Keywords
composition, constraints, ontology, web service.

1. INTRODUCTION
Traditional semantic web services composition approaches combine available services and order them to meet the user requirements. Most of them focus on services properties and work at a high level that cannot include user constraints. Although, exploring all possible solutions based on functional properties (Input, Output, Precondition, Effect) and Non-functional properties (Qos, etc) of the problem may not be feasible and some choices based on users constraints should be made to select the most appropriate services and to limit the search space [6,7].

User constraints can be separated in soft (preferences) and hard (strict) constraints. In this paper, we consider only hard user constraints and we will not address the issue of ordering services for selection; we will rather select and compose services in a way that match hard user constraints. These constraints could concern either services properties (e.g., the user wants to have a three stars hotel) or the choreography (e.g., the user wants the taxi service to travel from the airport to the hotel).

For services properties, constraints can be either local or global. Local constraints restrict the values of a particular attribute of single service. For example, stars.hotel=threeStar is a local constraint which restricts the value of a functional attribute of a single service. On the contrary, global constraints restrict simultaneously the values of two or more attributes of several services [10]. For example, location.Hotel=location.Restaurant c validRegion.Airport is a global dependent constraint which restrict simultaneously the value of many attributes for different services (hotel, restaurant, airport).

Choreography constraints are in the form of conditional constraints (like business rules); they express the order and the execution sequence of some activities imposed by the user (e.g., the user don’t take the bus or if the user is registered in hotel for services composition that: 

- Considers local, global and choreography user constraints.
- Considers dependencies between services in different layers: abstract layer and concrete layer.

Of course the requestor’s requirements must know the abstract services offerings in order to draw constraints. Many applications and research about composite web services emphasize on semi-automatic mode [1,2,3,4] by selecting from a list of abstract services, the best instances to be executed as mentioned in Fig.1. Most of existing approaches are syntax-based approaches based on techniques like Constraint Solver Problem [2,3,4,11] to match syntactically heterogeneous descriptions. However, considering global and choreography constraints between services requires to draw semantic relations between web services and their attributes.

Therefore our aim is to develop a semantic approach for web services composition that: 

- Considers local, global and choreography user constraints.
- Considers dependencies between services in different layers: abstract layer and concrete layer.

Figure 1. Web Service Composition Approach.
The rest of this paper is organized as follows. In the section 2, we classify some approaches that are related to our work. Section 3 proposes a framework and our goals. Finally, we summarize and conclude in section 4.

2. EXISTING APPROACHES
Over the past few years a number of approaches addressing the problem of the composition of web services have been proposed. In this section we present a number of criteria that we believe should be considered when comparing service composition research works that consider users constraints.

2.1 Services Properties
Services are described by their FP (Input, Output, Precondition, Effect) and their NFP(QoS). Dependencies between services must also be considered in composition in order to avoid the undesirable solutions.

2.2 Type of constraints
As indicated before, constraints could be either local or global. Global constraints, which restrict the values of many attributes of a multiple constituent services, can also be dependable or independent [10]. A global constraint is strictly dependent if the values that should be assigned to all remaining restricted attributes can be uniquely determined once a value is assigned to one (location.Hotel=location.Restaurant validRegion.Airport). Any global constraint that is not strictly dependent is independent [10]. (date_booking_flight<= date_booking_hotel<= date_booking_restaurant). User constraints can also consider some restrictions on the choreography in the conditional form like business rules.

2.3 Selection strategy
For selection authors use many strategies in instantiating abstract services by concrete services. In order to find the best solution that meets all criteria defined by user or by services properties, the instantiation is related to the use of tools based on constraint solving techniques [1,2,3,4], rules selection written in SWRL [5], Calcul event [6], etc. and scoring techniques like TLE[8,9] and ALC[10]. When authors [4,5,6] use local solver they also define backtracking and propagation process to find alternatives to some previously chosen node instantiation solution. Once the backtracking process execution terminates, resulting in newly chosen solution (instance), the composition solution must be computed and may require the propagation of the newly chosen solution to the higher node (in the hierarchical order).

2.4 Syntax versus semantic approach
Semantic-based approaches achieve high recall than syntax-based approaches. Attributes like: address, city, location are related semantically. By a semantic-based approach, we cannot only relate these concepts but also services like: bus, train and taxi, which are in the same category.

2.5 Online versus Offline approach
The choice of when composition happens is between offline (at design time) or online (during runtime). It depends on the use of dynamic attributes or static attributes. An attribute is static if its value is independent of the execution of the service.

2.6 Degree of automation
The ultimate goal of the composition research is to provide fully automatic processes. Such semi-automatic web service compositions are essentials when for example considering user constraints for each abstract service where specific web services are not predefined. All researches that we tackled are in semi-automated mode, but they differ in a gradual semi automatic mode or on-the-fly semi-automatic mode. In the first type, instances services are selected step by step and the result of each service are propagated in the rest of the hierarchies of the composition. The second mode starts by selecting the all candidate’s services and attributes and secondly by determining the solution.

2.7 Workflow structure
We distinguish between a static workflow (pre-defined) and dynamic workflow, where other modifications can affect it during execution. Also we distinguish between rich workflow that use many constructors (sequence, parallel (and), split (or), etc) and poor workflow that uses one or a few constructors. Especially, the split constructor decides the possible choice of workflow paths, which means that only when a service is selected and invoked, the next workflow path can be determined based on an assessment of the output data. However, the run time service output can not be predicted, as a result, an upfront selection has a chance to be completely wrong and would need to be recomputed after each split [8].

2.8 Summary
Table 1 summarizes the different studied works. They have in common that they are based on an existing abstract composition and in instantiating this abstract composition by selecting the most appropriate concrete web services with respect to constraints and by using one of the cited selection strategies. Some of them consider FP of services, other considers NFP or both and few of them consider dependencies between services. Composition can be done at design time or at run time according to the use of static or dynamic properties. Some authors [10], for reason of simplicity define only sequential operator in the workflow structure, others consider only local constraints [2,7,8,9,11]. None of these works handles simultaneously local, global, and choreography user constraints by a semantic approach that handles also the matching of heterogeneous descriptions of web services. Because most of the existing constraints solving techniques are purely syntactic (they match constraints based on their syntax) and because constraints on related attributes or related services are treated independently, we claim that a semantic approach to relate services and their attributes (before applying the constraints) is required. Thus, our goal is to address this open issue.

3. CONTRIBUTION
In order to develop a semantic approach for web service composition with constraints, we first introduce here a sort of conceptual framework based on three levels: communities, abstract and concrete, where:
Table 1. CLASSIFICATION OF THE DIFFERENT STUDIED WORKS

<table>
<thead>
<tr>
<th>Work</th>
<th>Services properties</th>
<th>Type of constraints</th>
<th>Selection strategies</th>
<th>Syntax/ semantic strategies</th>
<th>Degree of automation</th>
<th>Offline/ online</th>
<th>Workflow structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] NFP(Qos)</td>
<td>Local/Global</td>
<td>Global Solver</td>
<td>Syntax-based</td>
<td>On the fly</td>
<td>Offline</td>
<td>Static-rich</td>
<td></td>
</tr>
<tr>
<td>[2] FP</td>
<td>Local</td>
<td>Global Solver</td>
<td>Syntax-based</td>
<td>On the fly</td>
<td>Offline</td>
<td>Static-rich</td>
<td></td>
</tr>
<tr>
<td>[3, 4] NFP+ Static FP + dependencies</td>
<td>Local/Dependant global</td>
<td>Local Solver</td>
<td>Syntax-based</td>
<td>Gradual</td>
<td>Offline</td>
<td>Static-average</td>
<td></td>
</tr>
<tr>
<td>[8,9] NFP + dependencies</td>
<td>Local</td>
<td>Scoring Selection</td>
<td>Syntax-based</td>
<td>Gradual</td>
<td>Runtime</td>
<td>Dynamic-rich</td>
<td></td>
</tr>
</tbody>
</table>

- The communities ontology which represents the common understanding classification and dependencies between the abstract services.
- The abstract services describing the functional properties of web services.
- The set of available services, which are the actual concrete services to the abstract ontology described by using non-functional properties.
- The mapping for the available services to the community ontology and the abstract services are managed by agents, which control how concrete services join or leave consistently their abstract services.
- The client service request with constraints to be expressed by using the communities ontology for choreography constraints and abstract/concrete services for global/local constraints.

This sort of multi-layered architecture is commonly found in the literature [7,11]. For instance, Benslimane et al show in [11], how concrete services are mapped to abstract services. Medajhed and Bouguettaya [12] use a community to cater for an ontological organization of web services sharing the same domain of interest.

In our framework, the use of meta-ontology to mix hierarchy, dependencies and properties of services are our key points to develop a semantic approach that handles different types of constraints and takes into consideration services dependencies. This is illustrated in Fig.2.

The big steps of our composition based on semi automatic mode are:

- Specify user query: the abstract services and their constraints are in the form: (S₁,S₂, ...Sₙ; Cₛ): Where Sᵢ is an abstract service, Cs: a list of constraints on single Sᵢ or several Sᵢ.

- Locate candidate abstract services and their concrete services that conform to the constraints (candidate (S'₁), candidate (S'₂), ... .Candidate(S'ₙ)), Candidates services means that if the user selects travel services then all concrete services for Taxi, train, bus, etc (sub-services with respect to communities ontology) are added to candidates services. Also, in this step, choreography constraints are verified. If the user doesn’t take the bus, all concrete services for booking bus service will be eliminated from candidate services.

- Actualize the previous list by verifying services dependencies and user constraints choreography in order to obtain an optimized new ordered list: (cand_opt (S'₁), cand_opt (S'₂), ... cand_opt(S'ₙ)). If that list contains ambulance service and hospital service then only concrete hospital_ambulance and concrete hospital_service which work-with together are added.

- Identify restricted related attributes of candidate services and with respect to the ontology, replace them by the more general concept. Location, region, city, which are related by constraints will be substituted by the same attribute.

- Develop a semantic selection of the best instances, which tackle all constraints and services properties. Our selection uses the Semantic Query-enhanced Web Rule Language (SQWRL) to access to OWL ontologies. This language adds querying capabilities to SWRL [13] by providing primitives to select, count and perform other operations on the results of a SWRL rule. Our choice is motivated by the nature of some choreography constraints that have business rule form.
We are currently developing a composition algorithm that combines and unifies the results of each step.

4. CONCLUSION

There exist several works related to the use of constraint solving for web services composition, but none of all these works consider local/global/choreography user constraint in a semantic approach. Moreover, a few of them consider service dependencies. Our aim is to develop a framework based on three levels, which consider all these dimensions in order to simplify the semantic matching between services in a composition process.

A prototype is under development; real examples must also be defined to improve our proposition and to show how our framework can improve recall and precision than a framework where services are seen as individual services (not collected into an organized framework of communities).

5. REFERENCES


