Expert Systems with Applications 36 (2009) 11021–11029

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Composing semantic Web services under constraints

E. Karakoc, P. Senkul *

Middle East Technical University (METU), Department of Computer Engineering, ODTU Bilgisayar Muhendisligi Bolumu, B-110, 06531 Ankara, Turkey

ARTICLE INFO

Keywords:
Web service composition
Semantic Web services
Constraints
Constraint programming
Schedule

ABSTRACT

Web service composition is an inevitable aspect of Web services technology, which solves complex problems by combining available basic services and ordering them to best suit the problem requirements. Automatic composition gives us flexibility of selecting best candidate services at composition time satisfying QoS requirements; this would require the user to define constraints for selecting and composing candidate Web services. In this work, a Web service composition approach is presented in which a rich set of constraints can be defined on the composite service. The output of the framework is the schedule of Web service composition in which how and when services are executed is defined. The basic properties of the proposed approach is converting Web service composition problem into a constraint satisfaction problem in order to find the best solution that meets all criteria defined by user and providing semantic compatibility and composability during composition.

1. Introduction

The Web is evolving into a distributed device of computation from a collection of information resources (Curbera, Nagy, & Weerawarana, 2001; Rao & Su, 2004). The need for composing the existing Web services into more complex services is also increasing, since it can result into new and more useful solutions.

A composite Web service refers to a process consisting of collaborating, heterogeneous Web services. In dynamic composition, the Web services to be used for activities are selected just prior to execution according to the requirements provided by the user. Despite all efforts, Web service composition is still a highly complex task, and it is already beyond the human capability to deal with the whole process manually (Rao & Su, 2004). The complexity, in general, comes from the following sources:

- The number of services available over the Web increases dramatically, and one can expect to have a huge Web service repository to be searched.
- The Web is a dynamic environment; frequently, services are updated and new services are created. Thus the composition system needs to detect the updates on services just before the execution and the decision should be made based on the up-to-date information.
- On service composition, there can be many candidates for an abstract service, and it is necessary to select the correct Web service to achieve the composition goal and QoS requirements.
- There can be many constraints when selecting Web services or achieving composition constraints in a dynamic composition that defines how the services will be selected.

There are several works on Web service composition such as (Benatallah, Dumas, Fauvet, & Rabhi, 2002; Benatallah, Dumas, & Sheng, 2005; Lazovik, Aiello, & Gennari, 2005; McIlraith & Son, 2002; Medjadeh, Bouguettaya, & Elmagarmid, 2003; Rao & Su, 2004; Senkul, 2006; Sirin, Parsia, Wu, Hendler, & Nau, 2004). Most of the previous standards or methods on Web service composition concentrated on designing run-time behavior of the composition based on the pre-defined rules. However, generally in these methods, composition is totally pre-defined; Web services are available directly by URL. Therefore, when composition starts to execute, the goal of the composition may not be achieved due to wrong design of composition, violation of quality of service (QoS) constraints or non-available Web services. Those that aim dynamic composition generally focus on the service discovery without considering constraints on the composite service.

In this work, a constraint programming based approach for Web service composition problem is presented. This work focuses on above problems to generate solutions to find executable schedules by solving constraints. The main emphasized properties of the work are the ability to express and solve a rich set of constraints to model QoS requirements on the composite Web service and to provide a unified framework for this task. For modeling and satisfying user constraints, resource allocation management techniques of workflows are adapted for service selection (Senkul, Kifer, & Toroslu, 2002; Senkul & Toroslu, 2005). The proposed approach is realized in a Web service composition framework named composite Web service framework (CWSF). Using this framework, Web

* Corresponding author. Tel.: +90 312 2105518; fax: +90 312 2105544.
E-mail addresses: erman.karakoc@ceng.metu.edu.tr (E. Karakoc), senkul@ceng.metu.edu.tr (P. Senkul).

0957-4174/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.02.098

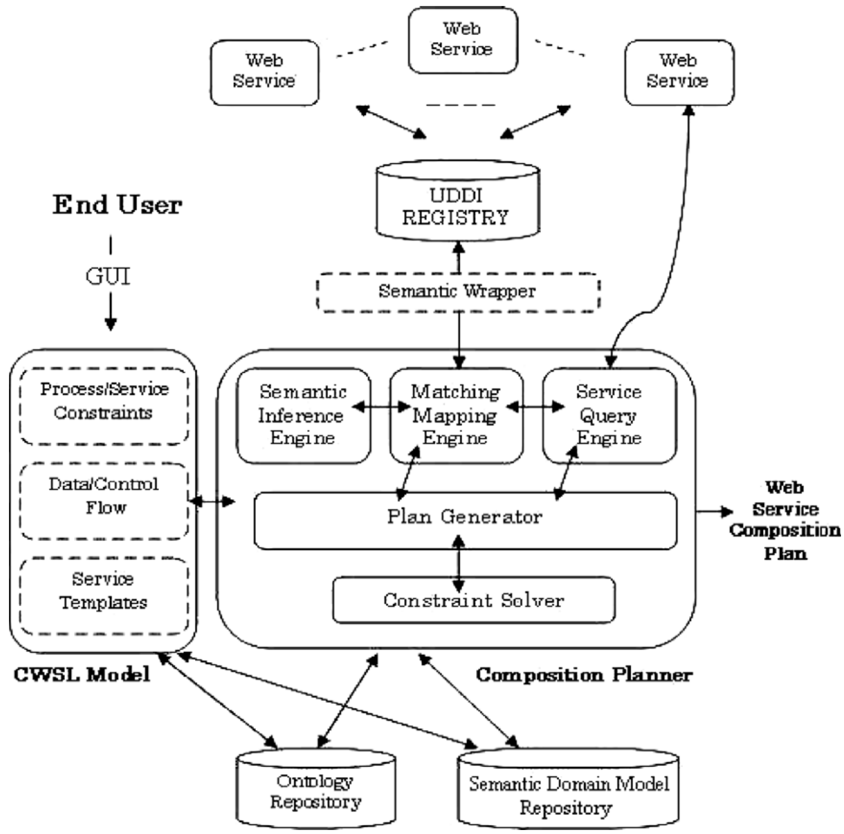

Fig. 1. Architecture of CWSF.

service composition problem is converted to a constraints satisfaction problem and composition plans are generated by utilizing a constraint solver. The output of the framework is the scheduling of Web service composition in which how and when Web services executed are defined.

Another important aspect of CWSF is using semantic information of the Web services for composition. Semantic matching techniques are utilized for the service discovery, service selection and composability control tasks of the composition process. By this way, selection of services that may satisfy the constraints does not rely on syntactic features, but uses the semantic model of the services.

The scope of this paper covers the general architecture of CWSF, modeling composite service under constraints and plan generation under constraints. For the details of the semantic service discovery and semantic composability checking tasks, the reader may refer to Kardas and Senkul (2007).

This paper is organized as follows. Section 2 describes the problem with the motivation example. Section 3 presents the related work. Section 4 introduces the framework, CWSF, which realizes the proposed approach. Section 5 gives information on the composite service and constraint modeling. Section 6 describes plan generation. In Section 7, experimental results on the scheduling performance of the approach are presented. Finally, Section 8 presents conclusion and future work.

2. Motivating example

An example of a Web service composition problem would be a *Travel Planner* system that combines the Web services of flight booking, travel booking, accommodation booking, car rental, and free activity planning, which are to be executed according to a given flow structure that captures the ordering constraints among the services, such as *Car rental service selection should take place after selection of hotel.*

Hotel booking, travel vehicle reservation and car rental activities are mandatory activities that should be executed. Traveler can go to holiday location by plane or by train, therefore only one of them should be selected. In addition, she wants to make some free activities such as flying a balloon, or ski in winter. In addition, traveler may define the following constraints:

- My maximum budget is only 2500$ for my travel.
- I want to go to *Barcelona* for my holiday.
- I want to start my travel on August 22, 2008 and finish at August 26, 2008.
- Distance between hotel and city center should not be more than 15 km.
- Hotel should be minimum 4-star. For 4 days, I can pay maximum 500$ for 4-star hotel and 750$ for 5-star hotel.
- I can give extra 200$ if hotel is *Hilton*.
- I prefer traveling by plane if it is cheaper than train.
- For free activities, my maximum budget is 350$.
- I want to use a *Toyota* during my holiday, no other cars.
- Hotel and car rental should be provided by the same service provider (i.e., by the same company).
- My budget for car rental is maximum 350$.
- I want to go by plane if it costs less than 550$.
- If I go by train, travel should not last longer than 8 h.
- My balloon tour should last at least 1 h.

The goal of CWSF for *Travel Planner* composite service is the to successfully organize the travel by selecting the concrete services for the abstract service templates based on the users constraints.