Automating the Composition of Transactional Web Services

Frederic Montagut, Institut Eurecom and SAP Labs, France
Refik Molva, Institut Eurecom, France
Silvan Tecumseh Golega, Hasso-Plattner-Institut, Germany

ABSTRACT

Composite applications leveraging the functionalities offered by Web services today are the underpinnings of enterprise computing. However, current Web services composition systems make only use of functional requirements in the selection process of component Web services while transactional consistency is a crucial parameter of most business applications. The transactional challenges raised by the composition of Web services are twofold: integrating relaxed atomicity constraints at both design and composition time and coping with the dynamicity introduced by the service oriented computing paradigm. In this paper, we propose a new process to automate the design of transactional composite Web services. Our solution for Web services composition does not take into account functional requirements only but also transactional ones based on the acceptable termination states model. The resulting composite Web service is compliant with the consistency requirements expressed by business application designers and its execution can easily be coordinated using the coordination rules provided as an outcome of our approach. An implementation of our theoretical results augmenting an OWL-S matchmaker is further detailed as a proof of concept.

Keywords: composition; transactional requirements; termination states; Web services

INTRODUCTION

Web services composition has been gaining momentum over the last years as a means of leveraging the capabilities of simple operations to offer value-added services. Complex services, such as airline booking systems, can be designed as the aggregation of Web services offered by different organizations. As for all cross-organizational collaborative systems, the execution of composite services requires transactional properties so that the overall consistency of data modified during the process is ensured.

Yet, existing Web services composition systems appear to be limited when it comes to integrating at the composition phase, the consistency requirements defined by designers in addition to functional matchmaking. Composite Web services indeed require different transactional approaches than the ones developed for usual database systems (Elmagarmid, 1992; Greenfield, Fekete, Jang, & Kuo, 2003). The transactional challenges raised by the composition of Web services are twofold. First, like classical workflow systems, composite services raise
less stringent requirements for atomicity in that intermediate results produced by some components may be kept without rollback despite the failure to complete the overall execution of a composite service. Second, composite services are dynamic in that their components can be automatically selected at run-time based on specific requests. Existing approaches only offer means to validate transactional requirements once a composite Web service has been created (Bhiri et al. 2005) and do not address the integration of these requirements into the composite application building process.

In this article, we propose a systematic procedure to automate the design of transactional composite Web services. Given an abstract representation of a process wherein instances of services are not yet assigned to component functional tasks, our solution enables the selection of Web services not only according to functional needs but also based on transactional requirements. In this approach, transactional requirements are specified by designers using the acceptable termination states (ATS) model. The resulting composite Web service is compliant with the defined consistency requirements and its execution can be easily coordinated as our algorithm also provides coordination rules that can be integrated into a transactional coordination protocol. Besides, the theoretical results developed in our approach have been implemented as a proof of concept and integrated into an OWL-S (OWL Services Coalition, 2003) functional matchmaker providing it with transactional matchmaking capabilities.

The remainder of the article is organized as follows. Section 2 and 3 introduce the methodology of our approach and a motivating example, respectively. In section 4, the transactional model underpinning this work is outlined. In section 5 we provide details on the termination states of a composite Web service then in section 6 we describe how transactional requirements are formed based on the properties of the termination states. The transaction-aware composition process through which transactional composite Web services are designed is detailed in section 7 while the implementation of our results in an OWL-S based framework is presented in section 8. Finally, section 9 discusses related work and section 10 presents the conclusion.

**PRELIMINARY DEFINITIONS AND METHODOLOGY**

Consistency is a crucial aspect of composite services execution. In order to meet consistency requirements at early stages of the service composition process, we need to consider transactional requirements a concrete parameter determining the choice of the component Web services. In this section we present a high level definition of the consistency requirements and a methodology taking into account these requirements during the composition of Web services.

**Consistent Composite Web Services**

A composite Web service $W_s$ consists of a set of $n$ Web services $W_s = \{s_a\}_{a \in [1,n]}$ whose execution is managed according to a workflow $W$ which defines the execution order of a set of $n$ tasks $W_s = \{t_a\}_{a \in [1,n]}$ performed by these services (for the sake of simplicity, we consider in our approach that a given service executes only one task). The assignment of services to tasks is performed by means of composition engines based on functional requirements. Yet, the execution of a composite service may have to meet transactional requirements aiming at the overall assurance of consistency. Our goal is to design a service assignment procedure that takes into account the transactional requirements associated with $W$ in order to obtain a consistent instance $W_s$ of $W$ whose execution can be supported by a transactional protocol defined using these transactional requirements as depicted in Figure 1. We consider that each Web service component might fulfill a different set of transactional properties. For instance, a service can have the capability to compensate the effects of a given operation or to re-execute the operation after failure, whereas some other service does not have any of these capabilities.
Linked Data: Connecting Spatial Data Infrastructures and Volunteered Geographic Information
www.igi-global.com/chapter/linked-data-connecting-spatial-data/51488?camid=4v1a