Chapter 13
A Reengineering Approach for Ensuring Transactional Reliability of Composite Services

Sami Bhiri
National University of Ireland, and Galway Digital Enterprise Research Institute IDA Business Park, Ireland

Walid Gaaloul
TELECOM & Management SudParis, France

Claude Godart
LORIA-INRIA, France

ABSTRACT

Different from traditional software applications, Web services are defined independently from any execution context. Their consequent inherent autonomy and heterogeneity fostered by a continuous evolution in business context and requirements make ensuring the execution of a composite service as intended a challenging task. This chapter presents a reengineering approach to ensure transactional reliability of composite services. Contrary to previous approaches which check correctness properties based on the composition model, the authors start from executions’ log to improve services’ recovery mechanisms. Basically, the chapter proposes a set of mining techniques to discover the transactional behavior from an event based log. Then, based on this mining step, the authors use a set of rules in order to improve services’ reliability.

DOI: 10.4018/978-1-61520-684-1.ch013

INTRODUCTION

Nowadays, enterprises are able to outsource their internal business processes as services and make them accessible via the Web. Then, they can dynamically combine individual services to provide new added-value composite services (CS for short). Due to the inherent autonomy and heterogeneity of Web services, a fundamental problem concerns the guarantee of correct executions of a CS. An execution is correct if it reaches its objectives or fails (properly) according to the designers requirements.

Copyright © 2010, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Motivating example Let us consider an application for Online Travel Arrangement (OTA for short) carried out by a composite service as illustrated in Figure 1. The customer specifies its requirements in terms of destination and hotel through the CRS service. The application launches in parallel flight and hotel reservation (FR and HR respectively) (after a study of the local transport accommodations (LTA)). The ADC service disposes administrative documents. Then, the customer is requested to pay by credit card (PCC), by check (PCh), or by TIP (PTIP). The Send Documents (SD) service ensures the delivery of documents to the customer. To deal with exceptions, designers specify additional mechanisms for failure handling and recovery. First, they specify that the hotel reservation can be compensated (by cancellation for instance) when the service FR fails to book a flight, and reciprocally. Second, in order to ensure the payment they specify that the service PCh is a payment alternative to the service PCC. Similarly, they specify that the service PTIP is a payment alternative to the service PCh with the assumption that the service PTIP always succeeds. Finally, designers specify that CRS, LTA, ADC and SD services are sure to complete. The main problem at this stage is how to ensure that the specified composition model guaranties reliable executions.

Generally, previous approaches develop based on their modeling formalisms a set of techniques to analyze the composition model and check “correctness” properties. Although powerful, these approaches may fail in some cases to ensure reliable executions even if they validate the composition model formally. This is because properties specified in the studied composition model remains assumptions that may not coincide with reality.

Back to our example, let us suppose for instance that in reality (by observation of sufficient execution cases) FR and PCh services never fail and the service PTIP is not sure to complete. That means, among others, (i) there is no need for the service HR to support compensation policies (which can be costly), and (ii) the payment can fail while the hotel and flight reservations are maintained. Formal approaches cannot deal with such anomalies.

Mining the effective transactional behavior enables to detect these gaps and thereafter improve the application reliability. For instance, mining the transactional behavior enables to improve the OTA service composition model by specifying the service PCh as a payment alternative to the service PTIP (since we notice that PCh is sure to complete).

Overview of our approach As explained in section 2, we distinguish between the control flow and the transactional flow of a composite service. The former specifies a partial ordering between the executions of component services. The latter defines the recovery mechanisms. In this paper we present an approach to improve CS recovery mechanisms based on the analysis of their execution history. Figure 2 gives an overview of the main steps of our approach: