Chapter 4

Dynamic Maintenance of Service Orchestrations

Manel Fredj
INRIA Paris-Rocquencourt, France

Apostolos Zarras
University of Ioannina, Greece

Nikolaos Georgantas
INRIA Paris-Rocquencourt, France

Valérie Issarny
INRIA Paris-Rocquencourt, France

ABSTRACT

Service-oriented architectures evolved rapidly as the solution to the latest requirements for loosely-coupled distributed computing. Into this broad context several approaches emerged towards the discovery and the systematic composition/orchestration of services. One of the next challenges in this field is the maintenance of service-oriented architectures towards accomplishing the ultimate goal of constructing eternal service-oriented systems out of loosely-coupled basic engineering elements. The particular problem we deal with in this paper is the dynamic maintenance of service orchestrations in the presence of unavailable services. Specifically, we focus on the dynamic substitution of stateful services that become unavailable during the execution of service orchestrations. As an answer to this problem, we propose the SIROCO middleware platform based on ontology which is further detailed along with an experimental evaluation of our first prototype. Our findings show that SIROCO provides the necessary means for achieving dynamic maintenance with a reasonable expense on the execution of service orchestrations.

INTRODUCTION

Service Oriented Architecture (SOA) is an architectural style that emerged recently as the answer to the latest requirements for loosely-coupled distributed computing (Cardoso & Sheth, 2006). Inline with the conventional distributed computing paradigm, a functionality is decomposed into distinct architectural elements, distributed over the network. Nevertheless, in SOA the basic architectural elements (i.e., services) are by
themselves autonomous systems that have been developed independently from each other. Moreover, services evolve independently. A service may be deployed, or un-deployed at anytime. Its implementation, along with its interface may change without prior notification. Services are typically combined in a loosely-coupled manner by building service orchestrations. Basically, an orchestration is a workflow that consists of a set of activities which exchange data with a set of services. The orchestration incarnates the basic control and dataflow dependencies that govern the execution of these activities.

In the context of SOA, several research efforts grew with the main focus being on the discovery and the systematic composition/orchestration of services, e.g., (Ben Mokhtar et al., 2006; Berardi et al., 2005; Yang & Papazoglou, 2004). One of the next challenges in this field is the maintenance of service orchestrations towards accomplishing the ultimate goal of constructing eternal service-oriented systems out of loosely-coupled basic architectural elements (Fredj et al., 2008). To this end, in this chapter we focus on the dynamic maintenance of a set of executing orchestrations upon the unavailability of a service that is required for the execution of these orchestrations. To deal with this problem, we propose an approach that enables the dynamic substitution of the unavailable service with an available one. The proposed approach is aimed at W3C Web services (W3Ca, 2004); we assume that services exchange information with the rest of the world within SOAP messages; service interfaces are specified in SA-WSDL (W3Cb, 2007); finally, service orchestrations are specified in terms of BPEL (IBM, 2002). Dealing with the dynamic substitution of stateless services is more or less straightforward. Thus, we concentrate on the worst case that involves the dynamic substitution of stateful services. According to the standard WS-Resource Framework (OASIS, 2004), we assume that service state descriptions may be provided, along with service interface descriptions.

Several approaches that deal with the unavailability of services, e.g., (Salatge & Fabre, 2007), rely on the construction of fault tolerant service groups out of unreliable services. The formulation of fault-tolerant groups of services as proposed in the state of the art seems difficult to apply when considering that the constituent services may be offered by competitive organizations or businesses. In this realistic scenario no independent business (e.g., a hotel) will accept to register its online service as a passive backup member of a group of services. Similarly, no independent business will accept to register its online service in a group that realizes active replication, while knowing that this will involve devoting precious resources to the group without any actual benefit (many reservations made by the same customer to each of the active replicas, while only one of them will be validated at the end of the protocol that realizes the reservation process through the active replication group). Similarly, in the field of dynamic reconfiguration of conventional distributed systems, several approaches tackled the issue of substituting an entity for another prefabricated backup entity (Kramer & Magee, 1990; Goudarzi & Kramer, 1996; Hauptmann & Wasel, 1996; Minsky et al., 1996; Warren & Sommerville, 1996; Bidan et al., 1998; Blair et al., 2000; Poladian et al., 2004). As previously discussed, the problem of service substitution is far more complex. In SOA, we can assume the possible existence of several semantically compatible services capable of performing the same or similar tasks. However, each one of them constantly serves requests and can not be considered as a passive backup for other services.

Therefore, the service substitution process that we are after consists of (1) discovering candidate substitute services out of a set of semantically compatible services that can be used in place of a service, which becomes unavailable and, (2) trying to identify one amongst these candidates that can be used as an actual substitute; whenever possible the selected substitute service must be such that its