Chapter 14

An Integrated Framework for Web Services Orchestration

C. Boutrous Saab
Université Paris Dauphine, France

D. Coulibaly
Université Paris Dauphine, France

S. Haddad
LSV, ENS Cachan, CNRS, France

T. Melliti
Université d’Evry, France

P. Moreaux
Université de Savoie, France

S. Rampacek
Université de Bourgogne, France

ABSTRACT

Currently, Web services give place to active research and this is due both to industrial and theoretical factors. On one hand, Web services are essential as the design model of applications dedicated to the electronic business. On the other hand, this model aims to become one of the major formalisms for the design of distributed and cooperative applications in an open environment (the Internet). In this article, the authors will focus on two features of Web services. The first one concerns the interaction problem: given the interaction protocol of a Web service described in BPEL, how to generate the appropriate client? Their approach is based on a formal semantics for BPEL via process algebra and yields an algorithm which decides whether such a client exists and synthesize the description of this client as a (timed) automaton. The second one concerns the design process of a service. They propose a method which proceeds by two successive refinements: first the service is described via UML, then refined in a BPEL model and finally enlarged with JAVA code using JCSWL, a new language that we introduce here. Their solutions are integrated in a service development framework that will be presented in a synthetic way.

DOI: 10.4018/978-1-61350-104-7.ch014
INTRODUCTION

At the hour of fusions, reorganizations of companies, Information Systems (IS) must have the capacity to take into account these economic constraints. What results is a need for flexibility, adaptability, opening and even for interoperability between remote and/or heterogeneous IS, i.e. based on different technical bases. Interoperability supposes that the applications are able to be located, to be identified, to expose the functionalities (services) which they offer, and finally, to exchange data. The Web services arise today as the most suitable solution in order to connect remote IS, eventually heterogeneous ones.

Indeed, Service Oriented Architectures Services (SOA), initially based on the components and their capacity to communicate through their interfaces, quickly showed their limits in terms of weak coupling (necessary to meet the needs for flexibility and adaptability) and of interoperability. The Web services bring these properties to the SOA which were precisely lacking with the component based architectures. The Web services lie on standards for the information exchange as well as on protocols for their transport.

Web services are “self contained, self-describing modular applications that can be published, located, and invoked across the Web” (Tidwell, 2000). They are based on a set of independent open platform standards to reach a high level of acceptance. Web services framework is divided into three areas - communication protocol, service description, and service discovery - and specifications are being developed for each one: the “Simple Object Access Protocol” (SOAP) (Gudgin, 2000), which enables communication among Web Services, the “Universal Description, Discovery and Integration” (UDDI) (Bellwood, 2002), which is a registry of Web Services descriptions and the “Web Services Description Language” (WSDL) (Christensen, 2001), which provides a formal, computer-readable description of Web services. The latter describes such software components by an interface listing the collection of operations that are network accessible through standard XML messaging. This description contains all information that an application needs to invoke such as the message structure, the response structure and some binding information like the transport protocol, the port address, etc.

However, simple operation invocation is not sufficient for some kind of composite services. They also require a long-running interaction derived by an explicit process model. This kind of services may often be encountered in two cases. First, when a Web service is developed as an agent, it is composed of a set of accessible operations and a process model which schedules the invocation to a correct use of the service. Secondly, facing to the capability limits of Web services, composite services may be obtained by aggregating existing Web services in order to create more sophisticated services (and this in a recursive way).

In order to deal with the behavioural aspects of complex services, some industrial and academic specifications languages have been introduced. Among them, Business Process Execution Language for Web Services (BPEL4WS or more succinctly BPEL) has been proposed by leading actors of industry (BEA, IBM, and Microsoft) and has quickly become a standard (Alves, 2007). BPEL supports two different types of business processes (Juric, 2005). (i) Executable processes specify the exact details of business processes. They can be executed by an orchestration engine. (ii) Abstract business protocols specify the public message exchange between the client and the service. They do not include the internal details of process flows but are required in order, for the client, to correctly interact with the service.

Given the description of an executable process, its associated abstract protocol is obtained by an abstraction mechanism (which masks all the internal operations of the service). However, the issues raised by these two types of processes are very different. On the one hand, the specification of an executable process is close to the definition