INFORMATION SCIENCE PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com **ITB13632**

This chapter appears in the book, Enterprise Service Computing: From Concept to Deployment edited by Robin Qiu © 2007, Idea Group Inc.

Chapter XV

Concepts and Operations of Two Research Projects on Web Services and Context at Zayed University

Zakaria Maamar, Zayed University, UAE

Abstract

This chapter presents two research projects applying context in Web services. A Web service is an accessible application that other applications and humans can discover and invoke to satisfy multiple needs. While much of the work on Web services has up to now focused on low-level standards for publishing, discovering, and triggering Web services, several arguments back the importance of making Web services aware of their context. In the ConCWS project, the focus is on using context during Web-services composition, and in the ConPWS project, the focus is on using context during Web-services personalization. In both projects, various concepts are used such as software agents, conversations, and policies. For instance, software agents engage in conversations with their peers to agree on the Web services that participate in a composition. Agents' engagements are regulated using policies.

Introduction

With the latest development of information technologies, academia and industry communities are adopting Web services because of their integration capabilities (Papazoglou & Georgakopoulos, 2003). Indeed, Web services can connect business processes in a business-to-business fashion. This connection highlights the possibility of composing Web services into high-level business processes usually referred to as composite services. Composition primarily addresses a user's request that cannot be satisfied by any available Web service (called service in the rest of this document); in this situation, a composite service obtained by combining available Web services might be used.

A Web service presents the following properties (Benatallah, Sheng, & Dumas, 2003): They are independent as much as possible from specific platforms and computing paradigms, are primarily developed for interorganizational situations, and are easy to compose so that developing complex adapters for the needs of composition is not required. For composition purposes, a composite service is always associated with a specification, which describes among others the list of component Web services that take part in the composite service, the execution order of these component Web services, and the corrective strategies in case these component Web services raise exceptions. Different composition languages exist such as the business process execution language (Curbera, Khalaf, Mukhi, Tai, & Weerawarana, 2003) and Web services flow language (Leymann, 2001). The primary objective of these languages is to provide a high-level description of the composition process far away from any implementation concerns. The specification of composite services is also concerned with the semantics of information that the component Web services exchange (Sabou, Richards, & van Splunter, 2003). However, the semantic composition is outside this chapter's scope.

Despite the wide embracement of Web services, they still lack the capability that could propel them to the acceptance level that features traditional integration middleware such as common object request broker architecture (CORBA) and distributed component object model (DCOM). This lack of capability is primarily due to the trigger-response exchange pattern that is imposed on Web services and their interaction models with third parties. The compliance with this pattern means that a Web service has only to process the requests it receives, without, for example, considering its execution status or even questioning about the validity of these requests. However, there exist several situations that call for Web services' self-management so that the requirements of flexibility, autonomy, and stability are met. By flexibility, we mean the capacity of a Web service to adapt its behavior by selecting the appropriate operations that accommodate the ongoing situation in which it operates. By autonomy, we mean the capacity of a Web service to accept demands of participation in composite services, or to reject such demands in case