Chapter XVI
Service Composition Approaches for Ubiquitous and Pervasive Computing: A Survey

Mohamed Bakhouya
The George Washington University, USA

Jaafar Gaber
Universite de Technologie de Belfort-Montbeliard, France

ABSTRACT

This chapter describes and classifies service composition approaches according to ubiquitous and pervasive computing requirements. More precisely, because of the tremendous amount of research in this area, we present the state of the art in service composition and identify key issues related to the efficient implementation of service composition platforms in ubiquitous and pervasive computing environments.

INTRODUCTION

Ubiquitous and pervasive computing (UPC) are new paradigms with a goal to provide computing and communication services all the time and everywhere. In **ubiquitous computing** (UC), the objective is to provide users the ability to access services and resources all the time irrespective of their location (Gaber, 2000; Weiser, 1996). **Pervasive computing** (PC), often considered the same as **ubiquitous computing** in the literature, is a related concept that can be distinguished from **ubiquitous computing** in terms of environment conditions (Gaber, 2006). We can consider that the aim in UC is to provide any mobile device an access to available services in an existing network all the time and everywhere while the main objective in PC is to provide spontaneous services created on the fly by mobiles that interact by ad hoc connections (Gaber, 2000, 2006)

In **ubiquitous and pervasive computing**, **service composition** plays a fundamental role, and automation is essential to improve speed and efficiency of users' responses and benefits. **Service composition** is the act of taking several component products or services, and handling them together to meet the needs of a given user (Chakraborty, 2001). For example, in an online business process of reservation of air tickets, a reservation service carries out three distinct services: ticket availability check, credit card check, and updating the required database to reserve a ticket for the user. Therefore, these three services must be integrated together to serve numerous user requests (Oprescu, 2004).

In UPC, automatic **service composition** requires dealing with four major research issues: service matching and selection, scalability, fault tolerance, and adaptiveness. The service matching and selection is the first step in creating any composite service and requires a **service discovery** system. The role of **service discovery** system is to locate the service components that provide the functionality to be placed in the new service. It allows the selection of services providing functionalities (i.e., capabilities) that match the requested functionalities. More precisely, **service discovery** systems should also be able to find out all services conforming to a particular functionality, irrespective of the way of invocation. To achieve this requirement, semantic level reasoning of describing the functionality of service is required. This is why the **Web service** community has developed a number of languages to formally describe services in order to facilitate their discovery.

Recently, **Web services** are becoming the most predominant paradigm for distributed computing and electronic business. In other words, the Web has become the platform through which many companies communicate with their partners, interact with their back-end systems, and perform electronic commerce transactions. Examples of **Web services** include bill payment, customized online newspapers, or stock trading services. As pointed out in Hu (2003), a **Web service** is a software system designed to support interoperable machine-to-machine interaction over a network. More precisely, **Web services** are self-contained, modular units of application logic that provide business functionality to other applications via an Internet connection (Bucchiarone & Gnesi, 2006). Several XML-based standards are proposed to formalize the specification of **Web services** to allow their discovery, composition, and execution (Baget, Canaud, Euzenat, & Saïd-Hacid, 2003; Zeng, Benatallah, Ngu, Dumas, Kalagnanam, & Chang, 2004). These standards are primarily syntactical; **Web service** interfaces are like remote procedure call and the interaction protocols are manually written. On the other side, the **Semantic Web** community focuses on reasoning about Web services by explicitly declaring their preconditions and effects with terms precisely defined in ontologies.

The **service discovery** system should also be scalable across large-scale networks and adaptable to dynamic changes especially when services dynamically join and leave the network. To implement this process, most **service discovery** systems like SLP (Guttman, 1999), Jini (Jini, 2000; Robert, 2000), and SSDS (Xu, Nahrstedt, & Wichadakul, 2001) require that service components must be stored in component directories that can be accessed at runtime. These centralized systems cannot meet the requirements of both scalability and adaptability simultaneously. Several decentralized systems are proposed to address these issues. A survey and classification of **service discovery** systems proposed in the literature are presented in Bakhouya and Gaber (2006b).

Service coordination and management is the second issue to be addressed in automatic **service composition**. More precisely, composition platforms must have one or some brokers that coordinate and manage the different services

Related Content