Chapter 12
Middleware Technologies for Ubiquitous Computing

Noha Ibrahim
Université de Lyon, France

Frédéric Le Mouël
Université de Lyon, France

Stéphane Frénot
Université de Lyon, France

ABSTRACT

Middleware handles many important functionalities for ubiquitous computing. The authors distinguish various middleware technologies providing key elements for all applications' requirements such as discovery, adaptation/composition, context management, and management of ubiquitous applications. In this chapter, they propose a classification for some of the most employed ubiquitous middleware. The classification was established upon the challenges raised by ubiquitous computing — effective use of smart spaces, invisibility, and localized scalability - and upon how the various ubiquitous middleware respond to them in terms of interoperability, discoverability, location transparency, adaptability, context awareness, scalability, security, and autonomous management. This classification shows that if many middleware are mature enough and offer specific functionalities respecting the properties of ubiquity, a real lack is noticed in having an interoperable, autonomous and scalable middleware for the execution of ubiquitous applications. The development of the service-oriented paradigm, the semantics, the Web middleware, and the ambient intelligence shows the new trend the middleware research field is engaged in.

INTRODUCTION

Middleware are enabling technologies for the development, execution and interaction of applications. These software layers are standing between the operating systems and applications. They have evolved from simple beginnings — hiding network details from applications — into sophisticated systems that handle many important functionalities for distributed applications — providing support for distribution, heterogeneity and mobility. The evolution of middleware has been influenced by numerous developments and standards efforts. Various middleware paradigms were defined. We
cite the Remote Procedure Call middleware (RPC), the Message Oriented Middleware middleware (MOM), the Object Request Broker middleware (ORB), and the Service-Oriented Architecture middleware (SOA).

The term middleware first appeared in the late 1980s to describe network connection management software, but did not come into widespread use until the mid 1990s, when network technology had achieved sufficient penetration and visibility. By that time, middleware had evolved into a much richer set of paradigms and services, offered to build distributed applications more easily and in a more manageable way. The term was associated mainly with relational databases for many practitioners in the business world through the early 1990s. By the mid-1990s this was no longer the case. Concepts similar to today’s middleware previously went under the names of network operating systems, distributed operating systems and distributed computing environments. Systems such as Apollo’s Network Computing Architecture (NCA), Sun’s RPC standard, and the Open Software Foundation’s Distributed Computing Environment (DCE) are all examples of successful RPC-oriented middleware that has been used for significant production applications. At the same time these systems were being produced in the 1980s and early 1990s, significant research was occurring in the area of distributed objects. Distributed objects represented the confluence of two key areas of information technology: distributed systems and object-oriented design and programming. As a result of research on these and other distributed object systems, combined with an evolution of the Transaction Processing monitor (TP monitor) into the concept of an “object monitor,” Common Object Request Broker Architecture (CORBA) was created. The CORBA system is built on top of ORB, which encompasses the entire communication infrastructure necessary to identify and locate objects, handle connection management, and deliver data. At the same time, a specific class of middleware, MOM, that operates on the principles of message passing or message queuing appeared. The MOM middleware unlike RPC and object-orientation is an asynchronous form of communication, i.e. the sender does not block waiting for the recipient to participate in the exchange. While the evolution of distributed objects in the 1990s was marked by significant efforts to establish standards such as CORBA, the evolution of messaging oriented middleware was practically devoid of standards efforts.

In the late 1990s, the growing popularity of Java, the explosive growth of the World Wide Web (WWW), and lessons learned from CORBA and messaging were all combined to form the Java 2 Enterprise Edition (J2EE), a comprehensive component middleware platform. J2EE provides support for numerous application types, including distributed objects, components, web-based applications, and messaging systems. Throughout the development of CORBA, J2EE, and proprietary messaging systems, Microsoft was busy developing its Distributed Component Object Model (DCOM).

In 1999 and 2000, the Web’s influence on middleware started to become readily apparent with the publication of the initial version of Simple Object Access Protocol (SOAP). Initially dubbed SOAP was the result of trying to create a system-agnostic protocol that could be used over the Web and yet still interface easily to non-SOAP middleware, including CORBA, DCOM, J2EE, and messaging middleware systems. Given the 1990s “middleware wars” between J2EE, CORBA, and DCOM, and between RPC and messaging, the unified support for SOAP was indeed groundbreaking. At the time, it seemed that SOAP, Web Services and Service-Oriented Computing in general might finally provide the basis for broad industry agreement on middleware standards.