Orchestration of Middleware Services

Hervé Paulino[1], Paulo Cancela[2], and Tiago Franco[2]

[1] CITI / DI - FCT - Universidade Nova de Lisboa, Portugal
[2] Critical Software SA, Portugal

Abstract. In this paper we present OHMS, a platform that provides an easy and not resource consuming way of exposing a platform to the Web, thus enabling Web access, business-to-business interaction and service composition, by the means of orchestration.

1 Introduction

Many of the current Service-Oriented Architectures (SOA) are built on top of Distributed Object (DO) technologies, such as CORBA [1] or DCOM [2], that have not overcome two crucial aspects of today's businesses: to port the SOA concept to the World Wide Web and to provide interoperability across many different platforms, enabling business-to-business transactions. By tackling both these issues, the Web Service technology (WS) has become the current standard for the development of SOA infrastructures.

The porting of DO-based platforms to the WS technology is, however, a costly process that requires high investments of both time and money. Furthermore, there are performance issues at stake, the overhead introduced by the WS platform and language independence is not desired when it comes to the internals of a platform. On the other hand, moving to the WS world opens a new range of prospects, essentially motivated by: increased visibility; business-to-business interaction based on XML standards; and the use of service composition to deploy new services by composing platform and other Web available services.

This paper bridges these two worlds by presenting OHMS (OrcHestration of Middleware Services), a framework to support the Web exposure and composition, concretely the orchestration, of services originating from distinct DO middleware technologies.

2 The OHMS Platform

The main goals of OHMS are: (1) to have a platform-centric approach, i.e., to focus on the bridging of service-oriented DO platforms, rather than individual services; (2) avoid, at all cost, the need to alter the platform's implementation in order to be suitable for orchestration; and (3) to have a general solution that is not bound to any particular technology.

To achieve such goals we designed an architecture composed of two independent modules: the **name-service directory module** bridges a DO platform,

R. Meersman, P. Herrero, and T. Dillon (Eds.): OTM 2009 Workshops, LNCS 5872, pp. 1–3, 2009.

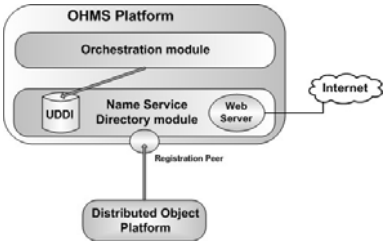

Fig. 1. Components of OHMS

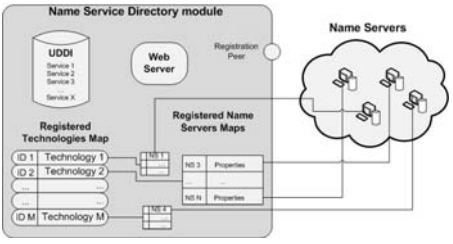

Fig. 2. The name-service directory

partially or completely, by storing information on its name-server and the logistic required to expose its service registry; and the **orchestration module** provides the means for the actual orchestration of the exposed services. It is an platform-centered extension of the Eclipse BPEL plug-in [3] that provides a simple way to access and orchestrate services of previously bridged platforms.

OHMS is not bound to any particular DO technology, thus providing full interoperability. From the directory's point of view, a DO technology is a set of Java classes that encapsulate all the logic necessary to bridge a service of the given technology: (1) inspect the registry of a platform's name-server, in order to extract the services to bridge; (2) generate the bridge for each of these services; and (3) register the resulting bridges in the UDDI registry, making them visible to the orchestration module.

The overall architecture of the directory, presented in Fig. 2, embeds a Web server, the access point to the Web services that relay the incoming invocations to the target platform services; and a UDDI registry that holds the registry of these Web services, publishing them to the network and serving as glue to bind both modules of the architecture. A registration peer provides the means to register, update and unregister both DO technologies and platforms.

By registering their name-server in OHMS, DO platforms automatically expose their set of services as Web services. The process is completely transparent to the platform, and thus no alterations on its implementation are required. The definition of which services to bridge is coded on a properties file supplied to the directory during the platform's registry.

3 Conclusions and Future Work

OHMS' directory is a compact engine that resorts to technology-dependent components, the bridging logic, to manage platform exposure. All of the work goes, thus, into developing the logistic from a given technology, something that must be done only once, and may be shared by the community. OHMS was validated in the context of COMCOP [4], a general purpose Command and Control (C & C) platform entirely developed in CORBA by Critical Software SA.

References

1. Object Management Group: The Common Object Request Broker: Architecture and Specification. Object Management Group (2001)
2. Horstmann, M., Kirtland, M.: DCOM Architecture. Microsoft (1997)
3. BPEL Project, `http://www.eclipse.org/bpel/`
4. Simoes, H., Carola, I., Franco, T.: C&C-Platform Reference Architecture. Technical report, Critical Software, S.A (2008)