Chapter VIII

A Method for Formulating and Architecting Component- and Service-Oriented Systems

Gerald Kotonya
Lancaster University, UK

John Hutchinson
Lancaster University, UK

Benoit Bloin
Lancaster University, UK

Abstract

This chapter describes a negotiation-driven method that can be used to formulate and design component and service-oriented systems. Component and service-oriented development are increasingly being promoted in literature as rapid low-cost strategies for implementing adaptable and extensible software systems. In reality, these development strategies carry significant risk throughout the system life cycle. The risks are related to the difficulty in mapping requirements to component and service-based architectures, the black-box software used to compose the systems, and the difficulty in managing and evolving the resulting systems. These problems underscore the need for software engineering methods that can balance aspects of requirements with
business concerns and the architectural assumptions and capabilities embodied in software components and services.

Introduction

The development of component and service-oriented systems share several characteristics (Bennett & Gold, 2003; Szyperski, 2002). Both approaches are based on development with reuse and are therefore constrained by the availability of suitable off-the-shelf software components and services. In both cases, negotiation is central to achieving a balanced solution. In both cases, the design of the interface is done such that the software component or service exposes a key part of its definition. In general functional terms, there is little difference to the consumer between reusing an existing internal component or service and buying or renting an external component or services. Differences arise in the nature of the applications and how they are composed (Szyperski, 2001).

Component-based development proceeds by composing software systems from prefabricated components (often third-party black-box software) (Brown & Wallnau, 1998; Szyperski, 2002). A typical component-based system architecture comprises a set of components that have been purposefully designed and structured to ensure that they fit together (that is, have pluggable interfaces) and have an acceptable match with a defined system context. Service-oriented development proceeds by integrating disparate heterogeneous software services from a range of providers (Cerami, 2002; Layzell et al., 2000; Stal, 2002). A service-oriented architecture is a means of designing software systems to provide services to either end-user applications or other services through published and discoverable interfaces. A typical service-oriented architecture comprises a service requestor, service provider, and service broker (registry) that interact through standard messaging protocols (for example, HTTP and SOAP) that support the publishing, discovery, and binding of services. However, the diverse nature of software systems means that it is unlikely that systems will be developed using a purely service or component-based approach (Kim, 2002; Kotonya & Rashid, 2001). Rather, a hybrid model of software development where components and services coexist in the same system is likely to emerge.

This chapter describes a method for software system development, COMPOSE (COMPonent-Oriented Software Engineering), that extends the notion of service to requirements definition to provide a framework for mapping requirements to hybrid component/service-oriented architectures. The method incorporates negotiation as a key process activity to balance aspects of system requirements and business concerns with the architectural assumptions and capabilities embodied in software components and services. The focus of the method is on system formulation and design. However, the method also provides hooks that allow it to be extended to system composition and management.