IRM PRESS
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.irm-press.com
ITB9329

Chapter XVI

A Service-Based Approach to Components for Effective Business-IT Alignment

Zoran Stojanovic
Delft University of Technology, The Netherlands

Ajantha Dahanayake
Delft University of Technology, The Netherlands

ABSTRACT

Although Component-Based Development (CBD) platforms and technologies, such as CORBA, COM+/.NET and Enterprise Java Beans (EJB), are now de facto standards for implementation and deployment of complex enterprise distributed systems, the full benefit of the component way of thinking has not yet been gained. Current CBD approaches and methods treat components mainly as binary-code implementation packages or as larger grained business objects in system analysis and design. Little attention has been paid to the potential of the component way of thinking in filling the gap between business and information technology (IT) issues. This chapter proposes a service-based approach to the component concept representing the point of convergence of business and technology concerns. The approach defines components as the main building blocks of business-driven service-based system architecture that provides effective business-IT alignment.

INTRODUCTION

The main challenges enterprises face today are how to manage complexity of systems being developed, effectively utilize the power of the Internet and be able to rapidly adapt to changes in both technology and business. The new paradigm of CBD has been introduced as an excellent solution for building complex Internet-enabled enterprise information systems (Szyperski, 1998; Brown & Wallnau, 1998). The basic idea of CBD originates from the strategy successfully applied in other engineering disciplines that a system developed from components is more flexible and easier to develop. CBD provides higher productivity in system development through reusability, more effective system maintenance, higher quality of solutions and the possibility for parallel work. Moreover, it provides better system adaptability through replaceability of parts, localization and better control of changes, system scalability and the possibility of using legacy assets.

The CBD paradigm has often been presented as a new silver bullet for complex, enterprise-scale system development in the Internet age (Udell, 1994). However CBD inherits many concepts and ideas from the earlier encapsulation and modularization — "divide-and-conquer" initiatives in information technology (IT). The NATO Conference in 1968 recognized that producing software systems should be treated as an engineering discipline providing system assembling from software components (McIlroy, 1968). Parnas (1972) defines concepts and requirements for decomposing system into modules. These principles of separation of concerns, encapsulation and plug-and-play building blocks have been applied in different ways through the concepts of functions, subroutines, modules, units, packages, subsystems, objects and now components.

The CBD paradigm was been first introduced at the level of implementation and deployment. CBD middleware technologies, such as CORBA Components (Siegel, 2000), Enterprise Java Beans (Sun Microsystems, 2002), and COM+/.NET (Microsoft, 2002), are now used as standards for the development of complex enterprise-distributed systems. While the technology solutions are necessary in building the system, one cannot simply program and deploy components using a component middleware, without any prior plan to follow from business requirements towards implementation. For the effective use of the CBD paradigm and in order to gain real benefits of it, the component way of thinking must be applied in earlier phases of the development lifecycle, such as system analysis and design. CBD methods and approaches proposed so far do not provide a complete and consistent support for various component concepts. Components are often treated as implementation concepts — packages of binary or source code that can be deployed over the network nodes. During the system analysis and design, components, if used, are often represented as larger grained business objects. This suggests using components mainly at the system implementation and deployment as software code packages, while still following the principles of object-oriented (OO) modeling, analysis and design.