The Evolution of Distributed
Component Architectures

Edward E. Cobb
BEA Systems, Inc.

Abstract. The growth of the Internet has coincided with the adoption of component system architectures. But early component architectures were built on assumptions that do not translate well when multiple business enterprises are involved. Nevertheless, these technologies have become widely adopted within the enterprise and form the basis for the new architectures that are emerging for business to business solutions. This paper looks at the evolution of component architectures in the past decade and tries to draw some conclusions about how these systems will evolve in the next several years.

1 Introduction

Business today is faced with tremendous competitive pressure to ensure survival. The information revolution, powered by the Internet, is changing the fundamentals of commerce which have prevail since the industrial revolution. For the first time in history, the customer, not the supplier, is in control of the value chain and that customer is demanding better and more personalized services. Expectations for services that will be “dial tone” quality --- always there and without noticeable delays, are likely to be the defining characteristic of the way business will be done in the 21st century. For businesses unable to adapt, there are competitors waiting --- ready and willing to do so, just a mouse-click away.

Companies like Dell and WalMart are pioneering this transformation, and they are taking market share from the established leaders in their respective markets and these companies are scrambling to avoid being left behind. With today’s corporation defined by its information systems, this has placed a premium on applications which can rapidly adapt to the forces of change. Component technology has emerged as the preferred solution for building these adaptive applications. Component architectures allow applications to be developed faster, enabling businesses to act more rapidly to gain and maintain competitive advantage.

2 Component Technology

Component technology is to software what manufacturing is to hardware. In hardware, the initial design is developed by highly-skilled engineers at a premium cost and takes considerable time to produce. Once it has been successfully proven however, it can be mass produced for a fraction of its original cost and be used in a multiplicity of products besides than the one it was originally designed for. Components promise the same kind of leverage for software development, enabling business applications to be constructed
from pre-built parts delivering higher quality solutions more rapidly using less skilled resources.

2.1 Components and Objects
The first software technology to address the productivity challenges of enterprise software development was object technology. Although object technology has its genesis in the late 1960s with languages like Simula, it entered the software mainstream in the late 1980s. The claims of its advocates were directly analogous to hardware development. Software objects would be fine-grained “chunks” of reusable code which could be put together in a variety of different ways to produce different applications. These objects would either be produced by the more experienced programmers in the customers IT organization or, in some future economy, would be available “off the shelf” from a variety of suppliers in the same way that computer system manufacturers like Compaq and HP buy chipsets from semiconductor manufacturers.

2.1.1 Objects on the Desktop
The first commercial success with object technology was on the desktop of the personal computer and workstation where it was used to build graphical end-user interfaces. This environment provided a perfect proving ground to apply the principles of object technology. The principal user of this new computing platform was the consumer, not the IT professional. Apple was the first to exploit this technology with its Macintosh operating system. The Macintosh was unique in offering only a graphical interface for end users. It had no command line interfaces like Unix®, PC DOS, and every operating system before it and its intuitiveness and simplicity made it an immediate success in the consumer marketplace.

The highly interactive end-user paradigm of the desktop demanded a systems designed with fine grain “chunks” of software to respond to mouse clicks and keyboard inputs. Both Microsoft and IBM responded to the success of the Macintosh by designing the next generation PC operating system, initially together but ultimately as competitive offerings. Microsoft’s Windows™ would ultimately prevail with its Component Object Model (COM™) technology and components, in the form of fine grain objects, would become the de facto standard for desktop development.

An important characteristic of the desktop programming paradigms is the tightly-coupled nature of the interactions between the various system objects providing the desktop functionality. A tightly-coupled architecture relies on a large amount of shared context. Because all the desktop objects executed on the same operating system, each “chunk” of software had a high degree of dependency on other “chunks” that it interacted with, not a surprise since they were all dealing with the same physical display. This design characteristic would carry over to early attempts at bringing the benefits of object technology to the server and the distributed systems environment.

2.1.2 Distributed Objects - CORBA and DCOM
In 1989, the Object Management Group (OMG) was founded with a charter to promote a single object-oriented applications integration environment based on appropriate industry standards. Over the years the organization has developed a variety of interoper-