Chapter IX

Understanding Distributed Object Oriented Systems

Alex Podaras
Hewlett Packard, USA

Distributed objects, as applied to the term distributed object oriented systems, can be defined as those objects that have many locations on a system (network), but stemming from the way they interact with one another, appear to be coming from just one location (Taylor, 1996, p. 263). Obviously, this presents distributed-object oriented systems with design complexities because the hardware and software are not located in one place but, to the user, must look as though they are. These complexities can be well appreciated by looking at the literature. Montlick (1999) uses the very term complex in a chapter heading related to distributed object oriented systems. He attributes this building complexity to the fact that object oriented technology is in its infancy. Given that distributed object oriented systems are complex and in their infancy, it is hard to decipher a clear definition of distributed object oriented systems and the client/server (frontline computer/back-line computer) model or environment. Some such as Berson (1996) say that client/server computing is a form of distributed computing, while others such as Taylor (1996) say that client/server computing is different from distributed computing. Understanding the client/server environment adds to the complexity of understanding distributed object oriented systems.

The purpose of this chapter, then, is to provide an understanding of what distributed object oriented systems are, no matter how complex they may appear to be. To provide a foundation for this under-
standing, the “building block” evolutionary process leading to the development of distributed object oriented systems will be given first. To foster an understanding of the systems themselves, it will be shown that no matter how complex, for a system to be distributed object oriented, basically several key ingredients must be in place. Accordingly, it will be shown that, fundamentally, distributed object oriented systems must have two object oriented properties or characteristics: encapsulation (the ability to hide code from the user) and messages (the way objects communicate). Additionally, it will be shown that software components (objects) of the distributed object oriented systems must have certain inherent features. Aside from the two object oriented properties and the certain inherent features, any critical system must have the ability to keep its data in a consistent state. This is particularly important when concurrent (at the same time) transactions (a unit of work) are executed.

It was determined that because distributed object oriented systems are complex and in their infancy, in order to produce a basic definition and understanding of what they are, it would be necessary to analyze a cross-section of the current literature: i.e., information found in books, articles, journals, and Internet sources as well as information obtained from interviews with an IT expert.

**What We Already Know**

**Evolution: Host-Based Environment to Object oriented Model**

As was mentioned in the above text, there are at least two ways of defining a client/server model (structural design) or environment (surroundings). One way is to say that distributed computing and client/server are the same (Berson); the other is that they are two separate entities (Taylor). For the purpose of obtaining a clear definition and understanding of distributed object oriented systems, Berson’s (1996) client/server rationale is used in this section.

**Host-Based Model**

In the host-based environment there is one processing computer (host) with several “dumb” terminals attached to it. “Dumb” terminals do not have the ability to process application components, which are application fragments; they are only used to communicate with the host and display output to the user. Application components in