4. How to Overcome the Shortcomings of ActiveX Component Technology

As mentioned earlier, ActiveX component technology has several shortcomings. Here, I discuss these shortcomings and their solutions. Firstly, one of the biggest controversies about ActiveX component technology is security. An ActiveX component can go to the end-user's local hard disk and have full access to system files, which creates the possibility of a hacker writing a component that reads and writes directly to that hard disk. The next time the end-user visits the hacker's Website, the component could pass a virus to the hard disk or do some other irreparable harm to the end-user's machine (Hopwood, 1997; Hoque & Sharma, 1998). However, while the lack of a Java-like SandBox security mechanism in ActiveX component technology is a problem for Internet-based BISs, it is not as much of a problem for Intranet- or Extranet-based BISs. All client programs, application servers, and databases are located on a closed network, so security and trust relationships are not an issue.

Secondly, ActiveX component technology is too complex a Web component technology that is difficult and even frustrating to understand and use. Fortunately, a VB-like tool, such as VB, Delphi, and PowerBuilder,

abstracts complex ActiveX component technology and makes it easy to use. With VB-like tools, developers now have the ability to develop ActiveX component-based BISs as easily and quickly as they created WinBISs, hence using their previous expertise and tools.

Furthermore, ActiveX components are binary objects, so they tend to be pretty large. They can take 3-5 minutes to download to end-user machines. However, unlike Java applets, ActiveX components usually need to be downloaded only once. Still, for someone who has never been to a Website with ActiveX components, this waiting could be tedious. Additionally, as with other programs, when developers update their ActiveX component, end-users are again required to download it (Hoque & Sharma, 1998). One simple way to reduce the size of ActiveX components for faster download is file compression. The most effective way to minimize this speed bump is decomposition.

There are two types of decomposition. The first type of decomposition is to separate common code from application-specific code into shared dynamic-link libraries (DLLs). This method offers four benefits:

1. The ActiveX component is smaller since it contains only the code unique to it; the common code is in DLLs.
2. An end-user machine with several DLL-enabled ActiveX components

installed on it needs only a single copy of DLLs, which are shared by all the ActiveX components.

3. Only minimal DLLs will be necessary to download as, if the correct version already exists on the end-user machine, they will not be downloaded.

4. The download of DLLs can be avoided when the DLLs are installed on end-user machines in advance.

The second type of decomposition is partitioning a program into a set of smaller independent pieces from a functional perspective. Thereafter, each piece is made into a small ActiveX component rather than an entire program made into a massive ActiveX component. The benefits of using this method are:

1. The ActiveX component is much smaller, and its size is independent of program size and complexity.

2. The method enables incremental program deployment. Thus, each launch of the program requires only pertinent code, which includes one ActiveX component and its dependent DLLs, be downloaded and installed; non-pertinent code is delayed until its first use.

3. The method enables partial program update. When a program is

modified, some code may be independent of the changes and hence need not be downloaded and re-installed.

4. The method enhances the possibility of code reuse due to high cohesion and loose coupling features of ActiveX components.