Jesse Berst

G U E S T

OPINION

IMPLEMENT OLE 2.0

AND OLE

AUTOMATION

AT YOUR OWN PACE,

NOT MICROSOFT'S.

Jesse Berst is Editor and Publisher of the Windows Watcher newsletter. Contact him at 15127 NE 24th St., Suite 344, Redmond, Washington 98052; via MCI Mail: JBERST; or CompuServe at: 71337,2052.

Caution: OLE 2.0 Ahead

Everybody reading this magazine should support OLE 2.0. Some day. The question is not whether. It's when. And for many corporate and commercial programmers, that means a slow, cautious approach. The Microsoft evangelists don't always give enough airtime to OLE's pitfalls. I've noticed four key problems.

1. OLE DOESN'T WORK YET. Try creating a complex compound document on an 8MB 386 machine. Object linking and embedding is painfully slow and cumbersome on the typical corporate desktop computer. Even worse, when users open three or four big applications at once, they experience delays and crashes. Now try mailing around that compound document. Because OLE 2.0's link tracking is weak, the links probably break and the document has little value.

It gets worse. Windows has no standard document formats. If you create a compound document using six different applications, everyone who receives that document must own the same six applications—they probably need the exact same versions. Without those exact same applications, users can't view or print the linked and embedded objects. If you want to distribute these documents to your clients and suppliers, they, too, must own all six applications. So don't think that end users will flock to your product just because it has OLE 2.0 support. Some day, OLE 2.0 will be a mandatory feature. But not until it works better.

2. OLE IS STILL TOO HARD TO PROGRAM. Not only does OLE 2.0 introduce 400 new API calls, it also forces programmers to follow a different programming model. If you have an older app you may need to tear it down and start all over again to make OLE 2.0 work.

Part of the difficulty comes from the way Microsoft has pulled the rug out from under early adopters. Due to "bug fixes," the company has changed the spec several times, to the detriment of companies that supported the preliminary specs. Still, most of the trouble has nothing to do with Microsoft. Switching to a new programming model literally takes years. OLE 2.0 forces you to make that switch.

This problem will improve in time. I'm convinced that Microsoft will introduce OLE 2.0 aids for its programming tools. In the meantime, don't think that OLE 2.0 is something you can add in the course of a week or two. It's a fundamental shift in the way you program. As such, it requires a major, long-term commitment.

3. OLE HAS COMPETITION. When Microsoft created OLE 2.0, the company chose to ignore IBM's System Object Model and the standards proposed by the Object Management Group, an industry consortium. Although I consider it unlikely, it is possible that OLE 2.0 could get sidetracked by rival specifications that adhere to open standards. The biggest competition will probably come from the OpenDoc specification now in the works from WordPerfect Corp., Apple Computer Inc., and IBM. Personally, I'm betting on OLE 2.0, despite its problems.

The OpenDoc committee has promised to support OLE 2.0, so an investment in OLE 2.0 compatibility would not be wasted, even if OpenDoc ultimately becomes the preferred standard. However, you could also find yourself forced to switch from OLE 2.0 to a different object model. It's a risk you should be aware of.

4. THERE IS NO SUPPORT STRUCTURE FOR OLE. Microsoft likes to talk about the day when you'll be able to buy OLE 2.0 components and plug them together to create custom systems. Microsoft ignores the fact that there's no way to sell or support such applications. Some day, this stumbling block may be overcome by companies that act like systems integrators. They will collect, customize, and assemble components into custom solutions. These foundries may even pay developers on a royalty basis, following the model of the record business. Or perhaps we'll see mainstream suite vendors distributing catalogs of compatible OLE components. Electronic and CD-ROM distribution will keep down the costs.

Right now, it's best to think of OLE 2.0 as a better model for building your software. Over time, you should re-architect your software along object-oriented lines. Over time, you should use Microsoft's component object model to modularize your software and reduce your development costs. Over time, you should take advantage of the new tools and testing mechanisms becoming available to make OLE 2.0 support much, much simpler.

Yes, you should move toward full OLE 2.0 compatibility. But do it at your pace, not Microsoft's. For most of you, that means a cautious, step-by-step approach. n